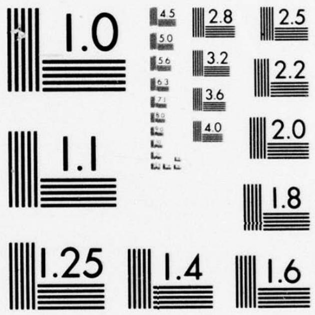
AD-A054 544 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
A POSTERIORI ERROR BOUNDS FOR TWO-POINT BOUNDARY VALUE PROBLEMS--ETC(U)
JAN 78 G KEDEM DAAG29-75-C-0024
UNCLASSIFIED MRC-TSR-1825 NL

| OF |
AD
A054544

END
DATE
FILMED
6-78
DDC


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(9) MRC Technical Summary Report #1825

(6) A POSTERIORI ERROR BOUNDS FOR TWO-POINT BOUNDARY VALUE PROBLEMS.

(10) Gershon Kedem

(14) MRC-TSR-1825

(15) DAAG29-75-C-0024

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706

(11) January 1978 (12) 74p.

(Received November 25, 1977)

Sponsored by

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

1473

221 200

LB

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

A POSTERIORI ERROR BOUNDS FOR TWO-POINT BOUNDARY VALUE PROBLEMS

Gershon Kedem

Technical Summary Report #1825
January 1978

ABSTRACT

Consider a general Two-Point Boundary Value Problem (TPBVP):

$$y'(t) = f(t,y) \qquad a \le t \le b$$

$$B_1 y(a) + B_2 y(b) = w \quad .$$

where $f: R^{n+1} \to R^n$, $f \in C^2$, B_1 and B_2 are $n \times n$ matrices and $w \in R^n$.

It is shown how one can bound a posteriori the error made in the numerical solution of the TPBVP. The error bounds obtained are rigorous and include the truncation and the roundoff error. In addition, the computations establish the existence of solutions to the TPBVP.

Numerical schemes are developed for the case where $f(t,y)$ is a polynomial in t and y. Examples are given of computational existence proofs for problems where analytical existence proofs are not known.

Key Words: Two-Point boundary value problem, a posteriori, error bounds

AMS(MOS) Subject Classification 65L10

Work Unit No. 7 -Numerical Analysis

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

SIGNIFICANCE AND EXPLANATION

Consider a general two-point boundary value problem

$$\begin{aligned} &y'(t) = f(t,y(t)) \\ &B_1 y(a) + B_2 y(b) = w \end{aligned} \qquad a \le t \le b \qquad (1)$$

where $y(t)$ and $f(t,y(t))$ are n dimensional vector-valued functions, B_1 and B_2 are $n \times n$ matrices and w is n vector.

Many problems in physics, chemistry and engineering can be put into the above form. In most cases, it is impossible to write down explicit formulas for solutions of two-point boundary value problems. Moreover, many times it is impossible to assert the existence of a solution or to know whether the equations have one, many or an infinite number of solutions.

Although there are many numerical methods that enable one to compute numerical approximations to solutions of equation (1), it is, in general impossible to tell a priori how close the numerical solution is to a true solution.

The literature holds examples of two-point boundary value problems that do not have a solution but the equations arising from the numerical approximation scheme do have solutions. Also, there are examples of numerical solutions that at first were believed to be good approximations but later were proved to be wrong.

We developed a numerical scheme that enables one to take a numerical solution to equation (1) and with the aid of that solution to compute, a posteriori, guaranteed error bounds. That is, one can compute how far the numerical solution is from the true solution.

The method described in this report is applicable in the cases where $f(t,y)$ is a polynomial in t and y. Although this is a serious restriction, many problems that arise in applications do fall into that category. We also give some indication of how one would extend our procedure to more general types of two-point boundary value problems, that is, cases where $f(t,y)$ is not a polynomial in t and y.

CHAPTER 1

A POSTERIORI ERROR BOUNDS FOR TWO-POINT BOUNDARY VALUE PROBLEMS

1.1 Introduction

Consider a general Two-Point Boundary Value Problem (TPBVP)

$$\begin{aligned} y'(t) &= f(t,y(t)) \qquad 0 \le t \le 1 \\ B_1 y(0) + B_2 y(1) &= w \end{aligned} \tag{1}$$

where $f:R^{n+1} \to R^n$, $f \in C^2$ and B_1, B_2 are $n\times n$ matrices, $w \in R^n$.

Unlike Initial Value Problems where the theory assures us that for a large class of equations there is a unique local solution, TPBVP can have one or many or no solutions at all. Unless one makes very strong assumptions on the function f in equation (1) the question whether a solution does exist or whether it is unique is hard to answer. Many times numerical solutions to TPBVP are computed without establishing the existence of a solution. The error analysis of numerical methods for solving TPBVP assures us that if a solution does exist and is locally unique and the discretization parameters are small enough, there is a locally unique numerical solution which is close to the true solution (see Keller [22], de Boor and Swartz [2], Krasnoselskii et al [24]). The above theory is unsatisfactory for two reasons: a) It is hard if not impossible to prove by analytic methods that there is a solution. b) In general it is impossible to know how small to make the discretization parameters.

Several numerical methods for estimating the error made in the numerical solution of differential equations are known. The most

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

famous one is Richardson Extrapolation to the Limit approach: If the equations and the solution are smooth enough, one can show that the error admits an expansion of the form

$$y_n - y(x_n) = \sum_{j=p}^{k} h^j \ell_j(x_n) + O(h^{k+1}).$$

Based on this expansion one can get asymptotic error estimates and one can improve the solution. In [47], Pereyra gives a survey of such methods. Henrici (see [16]) points out that one can solve numerically the differential equation satisfied by the dominant error term in order to get a good estimate of the error. Some methods like the Runga-Kutta-Fehlberg methods have error estimates built in. In [46], Zadunaiski describes another type of method. He interpolates the numerical solution by a local polynomial. Then, using that polynomial, he constructs a pseudo-problem for which the solution is known. He estimates the error in the numerical solution by computing the error made by the integration process in solving the pseudo-problem. More recently Babuska and Rheinboldt [1] analyzed some aposteriori error estimates for TPBVP of elliptic type solved by the finite element method.

Numerical estimates usually work very well. However, these are only estimates. Numerical estimates can always fail. It was pointed out by Lyness and Kaganove in their paper "Comments on the nature of Automatic Quadrature Routines" [25], that numerical schemes which estimate the error by considering function values at finitely many points inevitably may fail, even in the simplest case $y'(t) = f(t)$, that is integration.

Therefore, it is desirable to construct numerical schemes that will enable one to take a numerical solution and bound aposteriori the error in that solution. By doing so, one also proves existence of a solution. We would like to stress that we are not concerned with error estimates. We are interested in computing guaranteed error bounds. Error bounds are more expensive than error estimates, but in many cases the ability to talk about the accuracy of the numerical solution with certainty is important.

Many times the existence of a solution to a single second order Boundary-Value problem with separated boundary conditions can be established by the "Shooting Method": One can find two values of the missing initial condition such that the solution of the initial value problem with one initial value will "hit" above the end condition and the other below it. Since the solution of the differential equation is continuous as a function of the initial conditions (see Coddington and Levinson [8]), there must be a value of the missing initial conditions that will make the solution "hit" the right-hand side exactly. However, the above idea cannot be used to analyze more general problems.

A very general approach to the question of existence and aposteriori error analysis was suggested and analyzed by L. V. Kantorovich. He generalized Newton's method for solving nonlinear equations to a general method of solving nonlinear operator equations in Banach spaces.

In order to describe Newton's method in Banach space and Kantorovich's theorem we first describe the following concepts†. Let X and Y be Banach spaces and L be a bounded linear operator from X to Y. The induced operator norm of L is defined by

$$\| L \| = \sup_{\| x \|_X = 1} \| Lx \|_Y \ .$$

Let $B(X,Y)$ be the set of all bounded linear operators from X to Y. If we define addition and scalar multiplication in $B(X,Y)$ by

$$V = (W + Z) \Leftrightarrow Vx = Wx + Zx \quad \forall\, x \in X\ .$$

$$V = \alpha Z \Leftrightarrow Vx = \alpha Zx \quad \forall\, x \in X$$

then it is easy to see that V is a bounded linear operator from X to Y and that $B(X,Y)$ is a vector space. It is well known that $B(X,Y)$ is itself a Banach space. (see [20]).

Let P be an operator from an open set $D \subseteq X$ into Y. If there is a bounded linear operator $L \in B(X,Y)$ such that

$$\lim_{\| u \|_X \to 0} \frac{\| P(x_0+u) - P(x_0) - Lu \|_Y}{\|u\|_X} = 0,$$

then P is said to be differentiable at x_0 and the bounded linear operator $P'(x_0) = L$ is called the Frechet derivative of P at x_0. If P is differentiable at every point of the open set D then the mapping $P' : x_0 \mapsto P'(x_0)$ is an operator from $D \subseteq X$ to $B(X,Y)$ and therefore one can talk about the derivative of such an operator. That derivative is called the second Frechet derivative of P, at x_0, and is denoted by $P''(x_0)$.

†For a full treatment see Kantorovich and Akilov [20] and also Rall [32].

Note that $P''(x_0) \in B(X,B(X,Y))$ and for $u,v \in X$, $P''(x_0)[u] \in B(x,y)$ and $(P''(x_0)[u])[v] \in Y$.

The induced operator norm of $P''(x_0)$ is:

$$\| P''(x_0) \| = \sup_{\|u\|_X=1} \sup_{\|v\|_X=1} \|(P''(x_0)[u])[v]\|_Y$$

Example:

Equation (1) can be put in the above framework. Let X_1 be the space $(C^1[0,1])^n$ and $X_2 = (C\,[0,1])^n \times R^n$, that is:

$$X_1 = \{f | f_i \in C^1[0,1],\ i = 1,\dots,n\}$$

$$X_2 = \{\begin{bmatrix} f \\ \rho \end{bmatrix} | f_i \in C[0,1],\ \rho \in R^n\} .$$

A norm on X_1 is (for example)

$$\|f\|_{X_1} = \max_{1\le i\le n} [\max(\sup_{0\le t\le 1} |f_i(t)|,\ \sup_{0\le t\le 1} |f_i'(t)|)] .$$

A norm on X_2 is (for example)

$$\max(\max_{1\le i\le n} [\sup_{0\le t\le 1} (|f_i(t)|)],\ \max_{1\le i\le n} |\rho_i|) .$$

It is not hard to show that with the above definitions X_1 and X_2 are Banach spaces.

Let $P : X_1 \to X_2$ be defined by:

$$P(x)(t) = \begin{pmatrix} x'(t) - f(t,x(t)) \\ B_1x(0) + B_2x(1) - w \end{pmatrix} \quad 0 \le t \le 1 .$$

Clearly solving for $P(x) = 0$ is equivalent to solving equation (1). $P'(x_0)$ is the linear operator defined by

$$P'(x_0)v(t) = \begin{pmatrix} v'(t) - f_x(t,x_0(t))v(t) \\ B_1v(0) + B_2v(1) \end{pmatrix},$$

where $f_x(t,x_0(t))$ is the matrix

$$\left(\frac{\partial f_i(t,x_0(t))}{\partial x_j} \right)_{i,j=1}^{n}, \quad 0 \le t \le 1 .$$

The second Frechet derivative is given by:

$$P''(x_0)[v,u](t) = \begin{pmatrix} (f_{xx}(t,x_0(t))v(t))u(t) \\ 0 \end{pmatrix}, \quad 0 \le t \le 1$$

where $f_{xx}(t,x_0(t))$ is the tensor $\left(\frac{\partial^2 f_i(t,x_0(t))}{\partial x_j\, \partial x_k}\right)_{i,j,k=1}^{n}$. Note that the norm in X_1 involves the first derivative and $\|y-x\|$ is small only if $\sup\limits_{0\le t\le 1} |y(t)-x(t)|$ and $\sup\limits_{0\le t\le 1} |y'(t)-x'(t)|$ are small.

We are now ready to describe Newton's method in Banach space: Recall the regular Newton's method: One is trying to find x_* such that $f(x_*) = 0$. Assume that we have x_0 which is an approximation to x_*. We try to find x_1 which is a better approximation to x_*. We write,

$$0 = f(x_*) = f(x_0+\Delta x) = f(x_0) + f'(x_0)\Delta x + Q(\Delta x)$$

If one neglect the term $Q(\Delta x)$ and solve for Δx one gets $\Delta x=-f(x_0)/f'(x_0)$. Now if $x_1 = x_0 + \Delta x$ then $x_1 = x_0-f(x_0)/f'(x_0)$, or in general $x_{k+1}= x_k -f(x_k)/f'(x_k)$ which is Newton's iteration scheme.

Following the same reasoning one can generalize the above procedure to operator equations in Banach space. Let x_0 be an approximation to x_*. Assume that P is twice continuously differentiable. We write

$$0 = P(x_*) = P(x_0+\Delta x) = P(x_0) + P'(x_0)\Delta x + Q(\Delta x)\ .$$

By definition of Frechet derivative, $P'(x_0)$, $\lim\limits_{\|\Delta x\| \to 0} \frac{\|Q(\Delta x)\|}{\|\Delta x\|} = 0$.

So if $\|\Delta x\|$ is sufficiently small one can neglect the operator $Q(\Delta x)$ and solve for Δx,

$$\Delta x = -[P'(x_0)]^{-1}P(x_0),$$

assuming of course that $[P'(x_0]^{-1}$ exists. Again the general scheme is $x_{k+1} = x_k - [P'(x_k)]^{-1} P(x_k)$.

Analyzing the above scheme Kantorovich proved the following remarkable theorem:

Theorem: (Newton's method in Banach space).

Let X and Y be Banach spaces and let $P:D \subseteq X \to Y$ (D open set) be a nonlinear operator. Assume that $P \in C^2(X,Y)$. Let $x_0 \in X$ and assume the following:

a) $[P'(x_0)]^{-1}$ exists

b) $\| x_1-x_0 \| \leq \eta$ where $x_1 = x_0 - [P'x_0)]^{-1}P(x_0)$

c) $\| [P'(x_0)]^{-1} \| \leq B$

d) $\| P''(x) \| \leq K$ for $\| x-x_0 \| \leq r$

e) $h = \eta \cdot B \cdot K < \frac{1}{2}$

f) $r \geq r_0 = \frac{1-\sqrt{1-2h}}{h} \eta$

then $P(x) = 0$ has a unique solution x_* in the ball $\| x-x_0 \| \leq r_0$ and the iteration scheme $x_{k+1}= x_k - [P'(x_k)]^{-1}P(x_k)$ will converge to x_*

The above theorem enables one to prove existence by first computing an approximate solution, then by computing the relevant constants one can establish the existence of a solution nearby. In addition, estimate (f) gives bounds on the difference between the computed solution and the exact solution. Although the theory has been known for more than twenty years it has been used very little in order to establish the existence of solutions to continuous problems like TPBVP. After a moment of thought one realizes that the actual application of the above abstract results to TPBVP is not trivial. A careful formulation of the problem has to be made.

In order to compute a rigorous bound on the error one has to be able to:

a) Compute a reasonable bound on $\| [F'(x_0)]^{-1} \|$.

b) Compute a good bound on $\| x_1 - x_0 \|$.

c) Compute a bound on $\sup\limits_{\| x-x_0 \| < r} \| F''(x_0) \|$.

First, one has to choose the spaces X and Y . A natural choice is X_1 and X_2 of the example. However, since the norm in X_1 depends on the first derivative and one needs to make $\eta = \| x_1 - x_0 \|$ small, the above choice implies that $x_0'(t)$ has to be a good approximation to $x_*'(t)$. This is a very stringent requirement. Therefore, it is desirable to find a formulation and a space X where the norm of X depends on function values only. Even in that case bounding the norm requires bounding the range of functions; hardly a trivial matter. (See Moore [29], and the references there.) Second, computing a bound on $\| F'(x_0)^{-1} \|$ implies bounding of norm of the Green's function of a first order, linear, two-point boundary value problem. Theoretical bounds for such a problem are usually impossible to obtain so one has to devise a computational scheme for getting such a bound. Third, one has to compute η and K. This computation takes bounding the range of one and two dimensional functions on a finite domain. This is not an easy task. (See Moore [29] and the references there.) All of the above gets even more complicated by the fact that computers compute with only a finite number of digits and every arithmetic operation introduces some round off errors. If one wishes to <u>prove</u> existence of a solution one has to take that point into account.

The residuals will be very small functions that are the difference between almost equal but not small functions. Therefore, it is impossible to decide apriori how many digits one needs to use in order to be able to neglect the effect of round-off error.

Several authors have suggested various formulations and computational schemes that enable one to compute the constants of Kantorovich's theorem. However, so far, the methods that are suggested in the literature have some difficulties. Talbot [37] has constructed a method that enables one to use a modified version of Kantorovich's theorem to bound the error for single second order boundary value problem $y'' = f(t,y)$ where the right hand side does not depend on y'. In order to take the truncation and the round-off error into account Talbot uses Interval Analysis techniques (see Moore [30]). He computes an interval valued function that contains the exact Green's function in its range. By computing a bound on the range of the interval function, he is able to compute a bound on $\| F'(x_0)^{-1} \|$. He also uses interval techniques to compute bounds on $\| x_0 - x_1 \|$ and $\| F'' \|$. His approach enables him to compute guaranteed global bounds on the error. The major disadvantage of his method is that it requires large memory storage space. Also, as pointed out by Talbot himself, it is desirable to solve a more general class of problems. McCarthy and Tapia [26] have suggested a different approach. They convert a single, polynomial, second order, two-point boundary value problem into an equivalent Volterra equation and use a modified version of Newton's method, due to Tapia, to compute error bounds on the approximate solution.

Their method also suffers from some difficulties. The bound on $\|F'(x_0)^{-1}\|$ is given in terms of e^L where L is a bound on the norm of a certain function. This bound is very pessimistic. For $L = 20$, $e^L > 10^8$. Such a large bound makes the possibility of getting realistic error bounds, or error bounds at all, rather small. Also the authors do not consider the effects of round-off error at all.

Roberts and Shipman [34] have suggested to convert the problem of two-point boundary value problem to a set of algebraic equations by looking for the missing initial conditions. Then, they solve that set of equations by the "shooting method". However, they do not consider the error made by the process of solving the initial value problem and therefore, their approach cannot be used to get rigorous error bounds.

In [12] Cruickshank and Wright suggested a method for bounding $\|F'(x_0)^{-1}\|$ for 2mth order single TPBVP. They use the Green's function of the 2mth derivative plus boundary conditions to convert the TPBVP into an equivalent Fredholm integral equation of the second kind. They use Kantorovich's Theory of Approximation Methods (see [20], Ch. XIV), to relate the norm of $(I-k)^{-1}$ to the inverse of the matrix that one gets by using polynomial collocation method to solve the TPBVP. Their approach can also be used for systems of equations and piecewise polynomial collocation methods as well. However, their approach can lead to the inversion of very large matrices. Although the matrices themselves are band or block band matrices, their inverse is full. Also it is not always possible to find an explicit form of the Green's function that will enable one to convert the problem to an equivalent Fredholm integral equation.

Our method is to convert equation (1) into an equivalent Volterra equation. Then we derive an explicit formula for the Green's function of that Volterra equation. Using the formula we construct a numerical scheme to compute bounds on $\| F'(x_0)^{-1} \|$ and $\| x_1 - x_0 \|$ for function $f(t,y)$ which are polynomials in t and y. We are using only spaces of continuous functions.

This has two advantages:

a) We can use local polynomial interpolation in order to produce continuous functions as opposed to global C^1 piecewise polynomial functions.

b) Only the values of the computed solution have to be a good approximation to the true solution and not to the derivatives. Getting a good approximation to the solution and its derivative is of course harder than obtaining a good approximation only to the solution itself. The requirement that $f(t,y)$ in equation (1) be a polynomial in t and y is very restrictive. However, a large number of problems that arise in applications (most notably one dimensional cases of Naviar Stokes equations), fall into that category.

We also would like to note that our method can be made to work for functions $f(t,y)$ which are factorable functions of t and y. $f(t,y)$ can be approximated by Taylor series approximation and the error made by that approximation can be accounted for. However, we do not treat that case and we leave it for further study.

We conclude the introduction with an example of TPBVP that can be treated by our method. The example looks simple but is far from being trivial.

Consider the system of ordinary differential equations describing the steady flow of a fluid contained between two parallel, infinite plane disks which are rotating about a common axis

$$h^{iv} + hh''' + gg' = 0$$

$$g'' + hg' - h'g = 0$$

$$h(0) = h'(0) = h(1) = h'(1) = 0$$

$$g(0) = \Omega_0 \, , \quad g(1) = \Omega_1 \, .$$

The above problems have attracted considerable attention. In [27], McLeod gives a list of more than twenty references about work concerning the above problem. Although the physically interesting problem is when $|\Omega_0|$ and $|\Omega_1|$ are very large, existence proofs are known for a very limited set of values of (Ω_0,Ω_1), (see Elcrat [14]). Moreover extensive computations have been carried out and multiple solutions had been observed (see Wilson and Schryer [41] and the references there). However, the known theory cannot assert the existence of such multiple solutions nor could it rule them out.

One "nice' feature of the above equation is that it is quadratic in the unknown functions. Therefore, F'' is a constant tensor. Using our method we prove the existence of a solution for boundary conditions outside the range for which analytical proof is known. Actually, Kantorovich's theorem guarantees the existence of a solution for boundary values in some ball around the original boundary values.

1.2 The Theory:

In order to apply Kantorovich's Theorem to the problem

$$\begin{cases} y' = f(t,y) \\ B_1 y(0) + B_2 y(1) = w \end{cases} \qquad 0 \le t \le 1 \qquad (1)$$

we use the following formulation:

Let $C[0,1]^n$ denote the space of continuous functions from $[0,1]$ to R^n. Let $r \in R^n$. We use $|r|$ to denote $\max_{1 \le i \le n} |r_i|$. For $f \in C[0,1]^n$. we define the norm of f by:

$$\|f\| = \sup_{0 \le t \le 1} |f(t)| = \sup_{0 \le t \le 1} \max_{1 \le i \le n} (|f_i(t)|).$$

We denote by $C_0[0,1]^n$ the space

$$C_0[0,1]^n = \{g \in C[0,1]^n \mid g(0) = 0\} .$$

Clearly $C[0,1]^n$ and $C_0[0,1]^n$ with the norm $\|\cdot\|$ are Banach spaces. Let A be $n\times n$ matrix. We use $|A|$ to denote $\max_{1 \le i \le n} (\sum_{j=1}^{n} |a_{ij}|)$.

Let $W(t)$ be a $n\times n$ matrix valued function on $[0,1]$. We define the norm of W by

$$\|W\| = \sup_{0 \le t \le 1} |W(t)|.$$

Let f be a twice continuously differentiable function from R^{n+1} to R^n. Define $F:C[0,1]^n \to C_0[0,1]^n \times R^n$ by

$$Fu(t) = \begin{cases} u(t) - u(0) - \int_0^t f(s,u(s))ds \\ B_1 u(0) + B_2 u(1) - w \end{cases} \qquad . \qquad (2)$$

Clearly solving for $Fu = 0$ is equivalent to solving equation (1) .

$F'(x_0)$: the Frechet derivative of F at x_0 is given by:

$$F'(x_0)[v](t) = \begin{cases} v(t) - v(0) - \int_0^t f_x(s,x_0(s))v(s)ds \\ B_1 v(0) + B_2 v(1) \end{cases} \qquad (3)$$

where $f_x(s,x)$ is the matrix $\left\{\frac{\partial f_i}{\partial x_j}\right\}_{i,j=1}^{n}$. If $\eta \in C_0[0,1]^n \times R^n$, $\eta = \binom{r(t)}{\rho}$, then $v = F'(x_0)^{-1}[\eta]$ is the solution of

$$\begin{cases} v(t) - v(0) - \int_0^t f_x(s,x_0(s))v(s)ds = r(t) \\ B_1 v(0) + B_2 v(1) = \rho \end{cases} \tag{4}$$

$F''(x_0)$ the second Frechet derivative is given by:

$$F''(x_0)[v,w](t) = \begin{cases} \int_0^t (f_{xx}(s,x_0(s))u(s))w(s)\,ds \\ 0 \end{cases} \tag{5}$$

where $f_{xx}(t,x)$ is the tensor $\left\{\frac{\partial^2 f_i}{\partial x_j \partial x_k}\right\}$ $i,j,k = 1,\ldots,n$.

Let $x_0 \in C[0,1]^n$ be fixed and denote $f_x(s,x_0(s))$ by $A(s)$. $A(s)$ is a $n \times n$ matrix valued function.

Let $Y(t)$ be the $n \times n$ matrix that solves the following initial value problem:

$$\begin{cases} Y'(t) = A(t)Y(t) \\ Y(0) = I\,. \end{cases} \quad , \quad 0 \le t \le 1 \tag{6}$$

We are now ready to derive a formula for $F'(x_0)^{-1}$.

Theorem:

Let $R = B_1 + B_2 Y(1)$

$F'(x_0)^{-1}$ exists if and only if R is nonsingular and

$$\begin{aligned} F'(x_0)^{-1}\binom{r(t)}{\rho} &= Y(t)R^{-1}B_1\int_0^t Y^{-1}(s)A(s)r(s)ds \\ &+ Y(t)(R^{-1}B_1 - I)\int_t^1 Y^{-1}(s)A(s)r(s)ds \\ &- Y(t)R^{-1}(B_2\, r(1) + \rho) + r(t) \end{aligned} \tag{7}$$

Proof:

Since $A(t)$ is continuous on $[0,1]$ it is well known (see Coddington and Levinson [8]) that $Y(s)$ is nonsingular for all

$s \in [0,1]$. So equation (7) is well defined provided R^{-1} exists.
Let $w(t) = Y(t) \int_0^t Y^{-1}(s)A(s)r(s)ds + r(t)$, then, $w(0) = 0$ and

$$w(t) - w(0) - \int_0^t A(s)w(s)ds - r(t) =$$

$$Y(t) \int_0^t Y^{-1}(s)A(s)r(s)ds - \int_0^t A(s)Y(s) \int_0^s Y^{-1}(u)A(u)r(u)du\,ds$$

$$- \int_0^t A(s)r(s)ds =$$

(integrating the middle terms by parts)

$$= Y(t)\int_0^t Y^{-1}(s)A(s)r(s)ds - Y(t) \int_0^t Y^{-1}(u)A(u)r(u)du$$

$$+ \int_0^t Y(s)Y^{-1}(s)A(s)r(s)ds - \int_0^t A(s)r(s)ds = 0$$

therefore

$$w(t) - w(0) - \int_0^t A(s)w(s)ds = r(t) .$$

Since $Y(t) - Y(0) - \int_0^t A(s)Y(s)ds = 0$, for any $\alpha \in R^n$ $v(t) = w(t) + Y(t)\alpha$ is a solution to:

$$v(t) - v(0) - \int_0^t A(s)v(s)ds = r(t). \tag{8}$$

Clearly any solution to (8) is of the form $w(t) + Y(t)\alpha$ for some $\alpha \in R^n$. If v is a solution of equation (4) then

$$v(t) = Y(t) \int_0^t Y^{-1}(s)A(s)r(s)ds + Y(t)\alpha + r(t)$$

for some $\alpha \in R^n$. The condition

$B_1 v(0) + B_2 v(1) = \rho$ implies that

$$B_1\alpha + B_2[Y(1)\{ \int_0^1 Y^{-1}(s)A(s)r(s)ds + \alpha\} + r(1)] = \rho$$

hence

$$(B_1 + B_2\, Y(1))\alpha = \rho - B_2[Y(1) \int_0^1 Y^{-1}(s)A(s)r(s)ds + r(1)]$$

hence

$$R\alpha = \rho - B_2[Y(1) \int_0^1 Y^{-1}(s)A(s)r(s)ds + r(1)] \tag{9}$$

Therefore if R is nonsingular

$$\alpha = R^{-1}\{\rho - B_2[Y(1)\int_0^1 Y^{-1}(s)A(s)r(s)ds + r(1)]\} \tag{10}$$

and

$$v(t) = Y(t)\{\int_0^t Y^{-1}(s)A(s)r(s)ds + R^{-1}[\rho - B_2Y(1)\int_0^1 Y^{-1}(s)A(s)r(s)ds - B_2r(1)]\} + r(t)$$

$$= Y(t)R^{-1}\{(B_1+B_2Y(1))\int_0^t Y^{-1}(s)A(s)r(s)ds - B_2Y(1)\int_0^1 Y^{-1}(s)A(s)r(s)ds + \rho - B_2r(1)\} + r(t)$$

$$= Y(t)R^{-1}B_1\int_0^t Y^{-1}(s)A(s)r(s)ds - Y(t)R^{-1}B_2Y(1)\int_t^1 Y^{-1}(s)A(s)r(s)ds + Y(t)R^{-1}(\rho - B_2r(1)) + r(t)$$

$$= Y(t)R^{-1}B_1\int_0^t Y^{-1}(s)A(s)r(s)ds - Y(t)R^{-1}(R-B_1)\int_t^1 Y^{-1}(s)A(s)r(s)ds + Y(t)R^{-1}(\rho - B_2r(1)) + r(t)$$

$$= Y(t)R^{-1}B_1\int_0^t Y^{-1}(s)A(s)r(s)ds + Y(t)(R^{-1}B_1 - I)\int_t^1 Y^{-1}(s)A(s)r(s)ds + Y(t)R^{-1}(\rho - B_2r(1)) + r(t).$$

Conversely by virtue of the above computation, equation (9) has a unique solution only if R is nonsingular. Q.E.D.

Remarks:

1) If we replace $Y^{-1}(s)A(s)$ by $-(Y^{-1}(s))'$, we can rewrite equation (7) as:

$$F'(x)^{-1}\binom{r(t)}{\rho} = -Y(t)R^{-1}B_1\int_0^t (Y^{-1}(s))'r(s)ds - Y(t)(R^{-1}B_1 - I)\int_t^1 (Y^{-1}(s))'r(s)ds - Y(t)R^{-1}(B_2r(1)+\rho) + r(t). \tag{11}$$

2) Using the above equation we get

$$\| F'(x_o)^{-1}\| \leq \|Y\| \{(\int_0^1 |(Y^{-1}(s))'|ds)\max(|R^{-1}B_1|, |R^{-1}B_1 - I|) + |R^{-1}B_2| + |R^{-1}|\} + 1. \tag{12}$$

1.3 Bounding the Constants of Kantorovich's Theorem:

In order to construct an effective numerical scheme we assume that the function $f(t,y)$ in equation (1) is a polynomial in t and y.

We would like to note that all functions involved can be approximated by local Taylor series expansions and therefore the procedures we are about to describe can be used to compute the desired bounds even if f is not a polynomial. The difference between the exact equation and the local piecewise polynomial approximation can be accounted for. In the present work we do not investigate this possibility, and we leave it for further study. In any case as was pointed out before, many problems that arise in applications are polynomials in t and y.

In order to take into account the round-off error we are using interval arithmetic. The use of interval arithmetic at the present time is very expensive since one has to use simulated floating point arithmetic. The ratio of speed between the hardware floating point arithmetic and the interval arithmetic we are using is about 1:1000.

Although the scheme we are proposing does not require a large volume of computation, the use of interval arithmetic makes the method expensive to use. If one is not interested in actually proving the existence of a solution and one does not wish to take round-off error into account the use of double precision will suffice. We would like to point out that in computing $\| F''(x_0)^{-1}\|$ and $\| x_1-x_0\|$ we use interval arithmetic only in order to take the round-off error into account. Therefore, we do not get the pessimistic bounds that the use of interval arithmetic can lead to.

1.3.1 Bounding $\|F''(x)\|$.

If the function $f(t,y)$ in equation (1) is a polynomial in t and y and we choose our initial guess $x_0(t)$ to be a piecewise polynomial function, then $f_{xx}(t,x_0(t))$ will be a tensor of order 3 whose components are piecewise polynomial functions.

One way to compute K (a bound on $\sup\limits_{\|x-x_0\| \leq r} (\|f_{xx}(t,x(t))\|)$ is to bound the norm of $F_{xx}(t,x_0(t)-I)$ where F_{xx} is the interval extension of f_{xx} (see Moore [30]) and I is the interval $[-r,r]$. $[F_{xx}(t,x_0(t)-I)]_{i,j,k}$ is a piecewise polynomial interval function. In section (1.4) we describe how one can bound the norm of such functions. Another way to compute K is to bound $\|f_{xx}(t,x_0(t))\|$ and then to compute a crude bound on $\sup\limits_{\|x-x_0\| \leq r} \|f_{xxx}(t,x(t))\|$. Since each component of $f_{xxx}(t,x)$ is a polynomial in t and x it is not hard to compute a crude bound on $\|f_{xxx}(t,x(t))\|$.

1.3.2 Bounding $\|F'(x_0)^{-1}\|$.

Let $A(t)$ denote $f_x(t,x_0(t))$. Let $Y(t)$ be the matrix solution of the initial value problem

$$\begin{aligned} Y'(t) &= A(t) \cdot Y(t) \\ Y(0) &= I \end{aligned} \qquad 0 \leq t \leq 1$$

It is well known that $Y^{-1}(t)$ exists and is the solution of:

$$\begin{aligned} Z'(t) &= -Z(t)A(t) \\ Z(0) &= I . \end{aligned}$$

Let $V(t)$ and $W(t)$ be matrix valued functions approximating $Y(t)$ and $Y^{-1}(t)$ respectively. Assume $V(0) = W(0) = I$.
Let $E_1(t) = Y(t) - V(t)$, $E_2(t) = Y^{-1}(t) - W(t)$. E_1 satisfies the

equation

$$E_1(t) = \int_0^t A(s)E_1(s)ds + X_1(t) \tag{13a}$$

where $X_1(t) = I - V(t) + \int_0^t A(s)V(s)ds$ also

$$E_2(t) = -\int_0^t E_2(s)A(s)ds + X_2(t) \tag{13b}$$

where $X_2(t) = I-W(t)-\int_0^t W(s)A(s)ds$. Therefore

$$|E_i(t)| \le |X_i(t)| + \int_0^t |A(s)||E_i(s)|ds \qquad i = 1,2.$$

In order to bound the norm of E_i in terms of $\|X_i\|$ and $\|A\|$ one can use the following well known lemma (See [8] Chap. 1).

Lemma (Gronwall inequality).

Let φ,ψ,χ be real piecewise continuous functions on a real interval $I = [a,b]$, $\chi \ge 0$ on I and suppose that $\forall t \in I$

$$\varphi(t) \le \psi(t) + \int_a^t \chi(s)\varphi(s)ds.$$

Then

$$\varphi(t) \le \psi(t) + \int_0^t \chi(s)\psi(s) \exp\left(\int_s^t \chi(u)\, du\right)ds.$$

The lemma implies that

$$|E_i(t)| \le |X_i(t)| + \int_0^t|A(s)||X_i(s)| \exp\left(\int_s^t |A(u)|du\right)ds$$

$$\le \|X_i\| \{1 + \int_0^t|A(s)| \exp\left(\int_s^t|A(u)|du\right)ds\}$$

$$= \|X_i\| \cdot \{1 + \exp\left(\int_0^t|A(u)|du\right)\left(\int_0^t -\frac{d}{ds} \exp\left(-\int_0^s|A(u)|du\right)ds\right)\}$$

$$= \|X_i\| \cdot \{1 + \exp\left(\int_0^t|A(u)|du\right)\left(1 - \exp\left(-\int_0^t|A(u)|du\right)\right)\}$$

$$= \|X_i\| \cdot \exp\left(\int_0^t|A(u)|du\right) \le \|X_i\| \cdot \exp(\|A\| t)$$

The above estimate is very pessimistic and for $\|A\| = 20$, $e^{20} \ge 10^8$ so the error bound will be very large. However, we can use this estimate on small subintervals as initial bound for the error and then improve that bound, that is: for $u \le t \le u+h$

$$|E_i(t)| \le |E_i(u)| + |X_i(t) - X_i(u)| + \int_u^t |A(s)| \cdot |E_i(s)| ds$$

hence

$$\|E_i(\cdot)\| \le (|E_i(u)| + \|X_i(\cdot) - X_i(u)\|) \cdot e^{h \cdot \|A\|} \qquad (14)$$

where the norms are computed on the interval $[u,u+h]$. If h is small the above estimate is not a gross estimate.

<u>Improving the bounds on $\| E_i \|$.</u>

Since $E_i(t)$ satisfies equation (13), one has, by the theorem in section (1.2)

$$E_1(t) = Y(t) \int_0^t Y^{-1}(s)A(s)X_1(s)ds + X_1(t)$$

$$= (V(t) + E_1(t)) \int_0^t (W(s) + E_2(s))A(s)\ X_1(s)ds + X_1(t)$$

hence

$$|E_1(t)| \le (|V(t)| + |E_1(t)|) \cdot \int_0^t (|W(s)| + |E_2(s)|)|A(s)||X(s)|ds$$
$$+ |X_1(t)| \ .$$

Similarly

$$|E_2(t)| \le (|W(t)| + |E_2(t)|) \int_0^t (|V(s)| + |E_1(s)|)|A(s)||X_2(s)|ds$$
$$+ |X_2(t)| \ .$$

Using the above estimates we arrive at the following algorithm:

<u>Notation:</u>

Let $0 = x_1 < x_2 \ldots < x_{n+1} = 1$ be a division of the interval $[0,1]$.

$$h_i = x_{i+1} - x_i$$

$$a_i \ge \sup_{x_i \le t \le x_{i+1}} |A(t)|$$

$$r_i \ge \sup_{x_i \le t \le x_{i+1}} |X_1(t) - X(x_i)|$$

$$\hat{r}_i \geq \sup_{x_i \leq t \leq x_{i+1}} |X_2(t) - X_2(x_i)|$$

$$v_i \geq \sup_{x_i \leq t \leq x_{i+1}} |V(t)|$$

$$w_i \geq \sup_{x_i \leq t \leq x_{i+1}} |W(t)|$$

$$z_i \geq \sup |X_1(t)|$$

$$\hat{z}_i \geq \sup |X_2(t)| \ .$$

We want to compute $\ell_i, \hat{\ell}_i$ such that

$$\ell_i \geq \sup_{x_i \leq t \leq x_{i+1}} |E_1(t)|$$

$$\hat{\ell}_i \geq \sup_{x_i \leq t \leq x_{i+1}} |E_2(t)|$$

define $\ell_0 = \hat{\ell}_0 = 0$.

Then by estimate (14) if

$$\ell_i^0 = (\ell_{i-1} + r_i)e^{h_i a_i}$$

$$\hat{\ell}_i^0 = (\hat{\ell}_{i+1}^0 + \hat{r}_i)e^{h_i a_i}$$

then

$$\ell_i^0 \geq \sup_{x_i \leq t \leq x_{i+1}} |E_1(t)|$$

$$\hat{\ell}_i^0 \geq \sup_{x_i \leq t \leq x_i} |E_2(t)|$$

also if

$$\ell_i^1 = (v_i + \ell_i^0) \sum_{j=1}^{i-1} (w_j + \hat{\ell}_j) z_j \ a_j h_j$$

$$+ (v_i + \ell_i^0) \quad (w_i + \hat{\ell}_i^0) z_i \ a_i h_i + r_i$$

and

$$\hat{\ell}_i^1 = (w_i + \hat{\ell}_i^0) \sum_{j=1}^{i-1} (v_j + \ell_j)\hat{z}_j\, a_j h_j +$$

$$+ (w_i+\hat{\ell}_i^0)(v_i+\ell_i^0)\hat{z}_i\, a_i h_i + \hat{r}_i$$

then

$$\ell_i^1 \geq \sup_{x_i \leq t \leq x_{i+1}} |E_1(t)| \quad \text{and} \quad \hat{\ell}_i^1 \geq \sup_{x_i \leq t \leq x_{i+1}} |E_2(t)|.$$

So we set $\ell_i = \min(\ell_i^0, \ell_i^1)$ and $\hat{\ell}_i = \min(\hat{\ell}_i^0, \hat{\ell}_i^1)$. We also would like to bound $E_2'(t)$ because we need to compute a bound on $\int_0^1 |(Y^{-1}(s))'|ds$. Since $E_2'(t) = A(t)\, E_2(t) + X_2'(t)$ it follows that

$$\| E_2' \| \leq \| A \| \cdot \| E_2 \| + \| X_2' \|$$

The above estimates enable one to compute good bounds on $\| Y \|$ $\int_0^1 |(Y^{-1}(s))'|ds$. First one computes numerically the values of Y and Y^{-1} on a sequence of points. Then interpolate these points by a local polynomial. This local interpolation scheme will give V and W. Then using the above algorithm one can compute rigorous bounds on $\| Y \|$ and $\int_0^1 |(Y^{-1}(s))'|ds$.

1.3.3 Computing a Bound on $\| R^{-1} \|$.

In the previous section we showed how one can compute $\delta_1 \geq |V(1) - Y(1)|$. Let

$$D = B_1 + B_2 V(1)$$

then

$$|R-D| \leq |B_2|\delta_1 = \delta_2 .$$

If $|D^{-1}| \cdot \delta_2 < 1$ then R^{-1} exists and

a) $$|R^{-1}| \leq \frac{|D|}{1-|D|\cdot\delta_2}$$

b) $$|R^{-1}-D^{-1}| \leq \frac{|D|^2 \cdot \delta_2}{1-|D|\delta_2} .$$

In general it is impossible to compute D^{-1} exactly. Only a good approximation to D^{-1} can be computed. Let C be a matrix such that $|I-CD| = \delta_3 < 1$ then $|D^{-1}| \leq \frac{|C|}{1-\delta_3}$.

Remarks:

Computing D^{-1} by Gauss elimination might not yield a very good approximation to D^{-1}. However, if we compute a matrix C such that $|I-CD| < 1$, this approximation can be improved by Newton's method. The condition $|I-CD| < 1$ is enough to guarantee that the iteration will converge. Therefore, one can compute an approximate inverse accurate to machine precision.

1.3.4 Computing a bound on $\eta = \| x_1-x_0\|$.

An obvious bound on η is $\| F'(x_0)^{-1}\| \cdot \|F(x_0)\|$, that is, the norm of the Green's function times the residual. If the norm of the Green's function is not very large the above estimate is good enough. However, this is an overestimate and η is usually much smaller.

One way of obtaining better bounds on η is to use the explicit form of the Green's function in order to compute $x_1(t)$, that is: Let $v(t) = x_1(t) - x_0(t)$, $r(t) = x_0(t)-\int_0^t f(s,x_0(s))ds - x_0(0)$ then by the theorem in section (1.2)

$$v(t) = Y(t) \int_0^t (Y^{-1}(s))^1 \, r(s)ds + Y(t)\alpha + r(t)$$

where α is given by equation (10). Therefore,

$$v(t) = (V(t) + E_1(t))\{\int_0^t (W'(s)+E_2'(s))r(s)ds + \hat{\alpha} + E_\alpha\} + r(t)$$

$$= V(t)\ \{\int_0^t W'(s)r(s)ds + \hat{\alpha}\} + r(t)$$

$$+ E_1(t)\ \{\int_0^t Y^{-1}(s)'r(s)ds + \alpha\}$$

$$+ V(t)\ \{\int_0^t E_2'(s)r(s)ds + E_\alpha\}$$

where $\hat{\alpha}$ is the computed value of α and $E_\alpha = \alpha-\hat{\alpha}$. Therefore

$$\|v\| \leq \|V(\cdot)\int_0^{(\cdot)} W'(s)r(s)ds+\hat{\alpha}\| \tag{15}$$

$$+ \|E_1\|\ (\int_0^1 |Y^{-1}(s)'|ds \cdot \|r\| + \|\alpha\|) + \|V\|\ (\|E_2'\|\cdot\|r\| + |E_\alpha| + \|r\|$$

In order to compute α, one can use the following:

Let C be the computed approximation to R^{-1} and $E_C = R^{-1}-C$. By equation (10) (assuming $\rho=0$)

$$\alpha = -R^{-1}B_2Y(1)\int_0^1 Y^{-1}(s)'r(s)ds - R^{-1}B_2r(1) =$$

$$= -(R^{-1}B_1-I)\int_0^1 Y^{-1}(s)'r(s)ds - R^{-1}B_2\ r(1) =$$

$$= -\{[(C+E_C)B_1-I]\int_0^1(W'(s)+E_2'(s))r(s)ds + (C+E_C)B_2r(1)\}$$

$$= -\{(CB_1-I)\int_0^1 W'(s)r(s)ds + C\ B_2\ r(1)\}$$

$$-\{E_C\ B_1[\int_0^1 W'(s)r(s)ds + \int_0^1 E_2'(s)r(s)ds] + E_C\ B_2\ r(1)\}$$

$$= \hat{\alpha} + E_\alpha\ .$$

A bound on $|E_\alpha|$ is given by:

$$|E_\alpha| \leq |E_C|\cdot|B_1|(|\int_0^1 W'(s)r(s)ds| + \|E_2'\|\cdot\|r\|)$$

$$+ |E_C|\cdot|B_2\ r(1)| \tag{16}$$

Note that since V,W,r are all piecewise polynomial functions, v is a piecewise polynomial function and all the necessary computations can be carried out. The ways one can compute bounds on $|E_c|$, $\|E_1\|$, $\|E_2^1\|$ and $\int_0^1 |Y^{-1}(s)^1| ds$ are already described in the previous sections. Therefore, by using estimates (15) and (16) one can compute a bound on η.

Another possibility for computing η is to solve numerically the linear TPBVP:

$$v' = Av+r$$

$$B_1 v(0) + B_2 v(1) = 0$$

where $r(t) = f(t,x_0(t)) - x_0'(t)$ and $A(t) = f_x(t,x_0(t))$. A bound on $\| x_1 - x_0 \|$ will be $\eta = \|v\| + \|F'(x_0)^{-1}\| \cdot \|q\|$ where $q(t) = v(0) + \int_0^t A(s)v(s)ds - v(t)$.

In order to get a better bound than $\eta = \| F'(x_0)^{-1}\| \|F(x_0)\|$ one has to take the discretization parameter smaller than the one used in the original equation or use higher order method (or both). Then, hopefully, $\|q\|$ will be much smaller than $\| F(x_0)\|$. Since we have a bound on $\| F'(x_0)^{-1}\|$ we know how small $\| q \|$ has to be in order to get a realistic bound on η.

1.4 Bounding the Norm of Piecewise Polynomial Functions.

In order to compute bounds on the quantities described in the preceeding sections one needs procedures for bounding the norm of piecewise polynomial functions on a finite interval. The polynomials may have interval coefficients.

In this section we are considering the above problem. The discussion is somewhat elementary. But finding effective procedures for bounding the range of piecewise polynomial functions is important enough to warrant our consideration. We wish to find a good balance between speed and accuracy. We are only considering robust algorithms, that is: algorithms that are guaranteed not to give erroneous answers.

We begin by showing that bounding the range of interval polynomials can be accomplished by bounding the range of at most 4 polynomials.

Let $P(x) = \sum_{i=1}^{n} A_i x^i$ be a polynomial of degree n with interval coefficients. We wish to compute $\|P\|$ on the interval $[a,b]$.

Case i) $0 \leq a$.

In that case let $\bar{q}(x)$ and $\underline{q}(x)$ be defined by

$$\bar{q}(x) = \sum_{i=0}^{n} \sup(A_i)x^i$$

$$\underline{q}(x) = \sum_{i=0}^{n} \inf(A_i)x^i \ .$$

Clearly $\forall x \in [a,b] \quad \underline{q}(x) \leq P(x) \leq \bar{q}(x)$ and $\bar{q},\underline{q}$ are the best possible functions satisfying the above inequality.

Case ii) $b \le 0$

Define

$$\bar{q}(x) = \sum_{i=0}^{n} \sup\,((-1)^{i}A_{i})x^{i}$$

$$\underline{q}(x) = \sum_{i=0}^{n} \inf((-1)^{i}A^{i})x^{i}$$

again. $x \in [a,b]$ $\underline{q}(x) \le P(x) \le \bar{q}(x)$

Case iii) $a < 0 < b$

Define

$$\bar{q}(x) = \begin{cases} \sum_{i=0}^{n} \sup\,((-1)^{i}A_{i})x^{i} & a \le x \le 0 \\ \sum_{i=0}^{n} \sup(A_{i})x^{i} & 0 \le x \le b \end{cases}$$

$$\underline{q}(x) = \begin{cases} \sum_{i=0}^{n} \inf((-1)^{i}A_{i})x^{i} & a \le x \le 0 \\ \sum_{i=0}^{n} \inf\,(A_{i})x^{i} & 0 \le x \le b \end{cases}$$

Therefore, by bounding, at most, 4 polynomials, one can compute bounds on the norm of $P(x)$. In our programs all polynomials are defined on the interval $[0,h]$ where h is the size of the subinterval . Therefore, only case i) applies. In practice we resort to the above reduction only if the width of the coefficients exceed a prespecified tolerance.

1.4.1 Computing a "Rough" Bound on Piecewise Polynomial Function.

If the interval on which the polynomial pieces are defined are small and the piecewise polynomial function itself is not very small, one can use the following effective procedure:

Assume that the polynomial $p(x)$ is defined on the interval $[0,h]$.

a) Determine upper and lower bounds on $P'(x)$; $\bar{q} \geq P'(x) \geq \underline{q}$

$x \in [0,h]$.

One way to compute $\bar{q},\underline{q}$ is as follows:

if $P'(x) = \sum_{i=0}^{n-1} b_i x^i$ let $\tilde{b}_i = \max(0,b_i)$ $\underset{\sim}{b}_i = \min(0,b_i)$

and set

$$\bar{q} = b_0 + \sum_{i=1}^{n-1} \tilde{b}_i h^i , \quad \underline{q} = b_0 + \sum_{i=1}^{n-1} \underset{\sim}{b}_i h^i$$

b) if $P'(x) \geq 0$ or $P'(x) \leq 0$ determine $\|P\|$ by evaluating $P(x)$ at the appropriate end point.

c) If $\underline{q} < 0 < \bar{q}$, then set

$$\|P\| = \max(|P(0)+h\bar{q}|, |P(0)+h\underline{q}|) .$$

A better bound can be computed by making use of the value of $P(h)$.

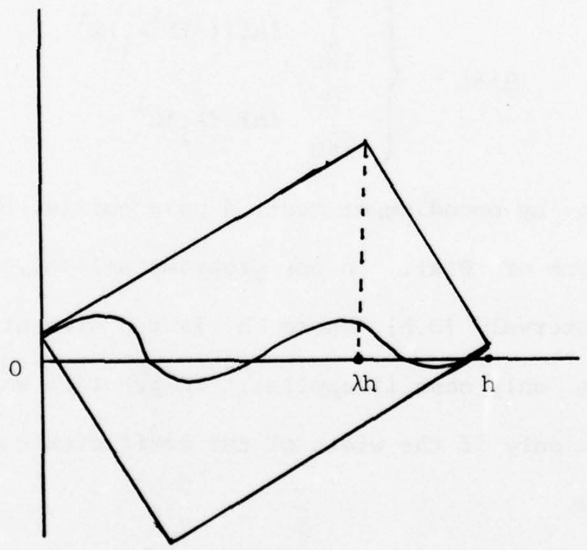


Figure 1.

Set $\|P\| = \max(|P(0)+\lambda h\bar{q}|, |P(0)+(1-\lambda)h\underline{q}|)$ where

$$\lambda = \frac{P(h) - P(0) - h\underline{q}}{h\bar{q} - h\underline{q}} .$$

It is our experience that the above procedure gives bounds with enough accuracy for bounding piecewise polynomial functions which are not small and the right order of accuracy of the bounds on the norms of piecewise polynomial functions that are very small.

We find the above procedure to be adequate for computing bounds on $\|A\|$, $\|V\|$, $\|W\|$ and for computing bounds on $\|W-I-\int_0 WAds\|$ and $\|V-I-\int_0 AVds\|$. The reason being that we only need to compute bounds that agree with the best possible bounds up to one or two digits and we do not need the best possible bounds. In many cases the above procedure is adequate for bounding

$$\|v(t) - v(0) - \int_0^t f(x,v(s))ds\| .$$

1.4.2 Computing a Good Bound on the Norm.

In order to compute a good bound on the norm of a polynomial $P(x)$, one can bracket the real zeros of $P'(x)$ inside very small intervals. Then one uses step c) in the previous method in order to bound the norm. One well-known method for bracketing the real zeros of a polynomial is the Sturm Sequences method (See for example Isaccson and Keller [19]). Another way to bracket is to use an algorithm suggested by Dussel and Schmitt [13]. First, one finds an integer r such that the rth derivative of P, $P^{(r)}$ does not have a zero in the interval. Using the fact that $P^{(r-1)}(x)$ can have at most 1 zero one can easily find out if $P^{(r-1)}$ does have a zero or not. If it does, the zero can easily be bracketed since $P^{(k-1)}$ is monotone. In the kth step, assuming that the zeros of $P^{(r-k)}(x)$ are bracketed, one can use the sign of $P^{(r-k)}(x)$ on each subinterval in order to bracket the zeros of $P^{(r-k-1)}(x)$.

We have chosen to follow the idea of Dussel and Schmitt, but instead of using an interval version of the Bisection Method as they do, we use an interval version of Modified Regula Falsi algorithm. (For details of Modified Regula Falsi algorithm see Conte and de Boor [9]). We use the above method only on intervals of which the function is monotone. In the case of multiple zeros we are using a simple gross bound. In that case the function is very flat and a good bound on the zeros of the derivative is not needed. A careful examination of all possible cases shows that the above procedure gives guaranteed lower and upper bounds for the zeros. The details of the above algorithm are given in the next section.

One last consideration. The intervals on which each polynomial is defined are small. Therefore, the contribution to the norm from the high powers is very small. One can bound the contribution from the low powers and then add the contribution from the absolute value of the high powers. This way the degree of the polynomials that one has to bound is reduced. Note that it is not essential to compute the bounds very accurately. Accuracy of two-three digits is good enough.

1.4.3 On Bounding the Range of a Polynomial.

Let $I = [a,b]$, $a \geq 0$, be a finite interval and $P(x) = \sum_{i=0}^{k} a_i x^i$ a kth degree polynomial. In order to bound the range of P on I we bracket the zeros of P', that is: We find a sequence of points $a \leq x_1 < x_2 \dots < x_n = b$ such that on each interval $[x_i, x_{i+1}]$ $P'(x) > 0$, $P'(x) < 0$ or $P'(x)$ may have a zero. We try to make the interval on which $P'(x)$ may have a zero as small as possible. Once we have bracketed the zeros of P' inside small intervals, we use the procedure described in the proceding section in order to bound the norm of the polynomial.

In order to bracket the zeros of P' we use recursively the following elementary fact (Rall's theorem).

Fact: If $f'(x)$ has no zeros in an interval, then f has at most one zero in that interval.

We start by finding an integer j as small as possible such that $P^{(j)}(x)$ has no zeros in I. Clearly there is at least one such j namely $j = k$.

If $j = 1$, then we are done. That is $P'(x)$ has no zeros in I. Otherwise, we assume we have a sequence of points $a = x_1 < x_2 < \dots < x_n = b$ and a list of integers $S_1, S_2, \dots, S_{n-1}$ such that $S_i \in \{-1,0,1\}$ and

If $S_i = 1$, $P^{(j)}(x) > 0$ on $[x_i, x_{i+1}]$.

If $S_i = -1$, $P^{(j)}(x) < 0$ on $[x_i, x_{i+1}]$.

If $S_i = 0$, $P^{(j)}(x)$ may have a zero in $[x_i, x_{i+1}]$.

We always try to make the intervals with possible zero smaller than some tolerance ε.

Given such a sequence and a list of integers, we compute a new sequence of points $a = y_1 < y_2 < \dots < y_m = b$ and a list of integers $V_1, V_2, \dots, V_{m-1}$ such that

$V_i = 1$ if $P^{(j-1)}(x) > 0$ on $[y_1, y_{i+1}]$

$V_i = -1$ if $P^{(j-1)}(x) < 0$ on $[y_i, y_{i+1}]$

$V_i = 0$ if $P^{(j-1)}$ may have a zero in $[y_i, y_{i+1}]$.

Clearly at most $k-1$ such steps are needed in order to trap the zeros of $P'(x)$ on I.

We now describe the procedure by which we construct the sequences $y_1, y_2, \ldots, y_m$; $V_1, V_2, \ldots, V_{m-1}$ with the aid of the sequences $x_1, x_2, \ldots, x_n$; $S_1, S_2, \ldots, S_{n-1}$. For convenience we denote $p^{(j-1)}$ by f and $p^{(j)}$ by f'.

We scan the interval I from left to right. On each subinterval $[x_i, x_{i+1}]$ we determine if f has a zero. If it does not we proceed to the next subinterval. If it does have a possible zero, then we bracket that zero in an interval as small as possible (up to a tolerance ε). In order to take round-off error into account, f is evaluated in Interval Arithmetic. So at any point x that we evaluate f we get two numbers $\bar{f}(x)$, $\underline{f}(x)$ such that $\bar{f}(x) \geq f(x) \geq \underline{f}(x)$.

One has to consider the following different cases:

<u>Case 1</u> $S_i = 1$.

In that case f is strictly increasing. Therefore, f has a zero in $[x_i, x_{i+1}]$ if and only if $f(x_i) \leq 0$ and $f(x_{i+1}) \geq 0$.

<u>Case 1.a</u>

If $\underline{f}(x_i) > 0$ or $\bar{f}(x_{i+1}) < 0$ f does not have a zero in $[x_i, x_{i+1}]$.

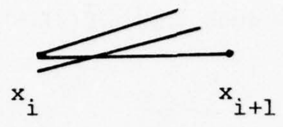
<u>Case 1.b</u>

If $\bar{f}(x_i) < 0$ and $\underline{f}(x_{i+1}) > 0$, then f has a zero in the open interval (x_i, x_{i+1}).

We can trap that zero inside a small interval by a disection method (Modified Regula Falsi) which we will describe later.

Case 1.c

$$\underline{f}(x_i) \leq 0 \quad \text{and} \quad \underline{f}(x_{i+1}) > 0 \quad \text{but} \quad \bar{f}(x_i) \geq 0$$



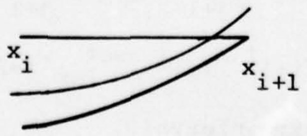
There are two different possibilities.

1) $i = 1$ in that case set $y_1 = x_1$ $V_1 = 0$. Since $\underline{f}(x_i) \leq 0$ and $\underline{f}(x_{i+1}) > 0$ we can use the disection method to find $\tilde{y}$ such that $\underline{f}(\tilde{y}) > 0$ and $\tilde{y}$ is as small as possible (that is: $\tilde{y}$ is a machine representable number and $\underline{f}(\tilde{y}-\varepsilon) \leq 0$). Set $y_2 = \tilde{y}$, $v_2 = 1$ and go to the next interval since f has no zeros in the interval $[\tilde{y}, x_{i+1}]$.

2) $i > 1$ In that case we have y_j such that f might have a zero to the right of y_j and $V_j = 0$. Again we can find $\tilde{y}$ as small as possible by the disection method such that $f(\tilde{y}) > 0$. Set $y_{j+1} = \tilde{y}$, $V_{j+1} = 1$ and go to the next interval. Note that if $\underline{f}(x_i) = 0$ we can take $\tilde{y} = x_i^+$, where x_i^+ is the smallest machine representable number greater than x_i because $\underline{f}(x_i) \leq f(x_j)$ and f is strictly increasing on $[x_i, x_{i+1}]$.

Case 1.d

$$\bar{f}(x_i) < 0 \quad \text{and} \quad \bar{f}(x_{i+1}) \geq 0 \quad \text{but} \quad \underline{f}(x_{i+1}) \leq 0$$



1) If $I = 1$ set $y_1 = x_1$, $V_1 = -1$ and find $\tilde{y}$ as large as possible such that $\bar{f}(\tilde{y}) < 0$. Set $y_2 = \tilde{y}$ and $V_2 = 0$.

2) If $i > 1$ we have y_j such that $\bar{f}(y_j) < 0$ and $V_j = -1$. Find $\tilde{y}$ as large as possible such that $\bar{f}(y) < 0$ and set $y_{j+1} = \tilde{y}$, $v_{j+1} = 0$.

Case 1.e

$$\underline{f}(x_i) \leq 0 \leq \bar{f}(x_i) \quad \text{and} \quad \underline{f}(x_{i+1}) \leq 0 \leq \bar{f}(x_{i+1}).$$

If $i = 1$ then set $y_1 = x_1$, $v_1 = 0$ and go to the next interval. If $i > 0$ then go to the next interval. (Remark: this is a very unlikely possibility).

Case 2 $S_i = -1$.

This case is almost the same as Case 1. The only difference is that the function is strictly decreasing.

Case 3 $S_i = 0$.

Case 3.a

There is no possible zero in the previous interval or $i = 1$.

In that case we compute $f([x_i,x_{i+1}])$ by interval arithmetic. If there is no zero we continue with the next interval. If there is a zero in $f([x_i,x_{i+1}])$ then

If $i = 1$ then $y_i = x_1$, $V_1 = 0$ and if $0 \notin [\underline{f}(x_2), \bar{f}(x_2)]$ set $y_2 = x_2$, $V_2 = \text{sign}(f(x_2))$. Otherwise go to the next interval.

If $i > 0$ set $y_{j+1} = x_i$, $V_{j+1} = 0$.

If $0 \notin [\underline{f}(x_{i+1}\bar{f}(x_{i+1}]$ set $y_{j+2} = x_{i+1}$, $V_{j+2} = \text{sign}(f(x_{i+1}))$ else go to the next interval.

Case 3.b

There is a possible zero in the previous interval.

If $0 \notin [\underline{f}(x_{i+1}), \overline{f}(x_{i+1})]$ set $y_{j+1} = x_{i+1}$ and $V_{j+1} =$ $\mathrm{sign}(f(x_{i+1}))$, otherwise go to the next interval.

Remark: The above procedure does not give the tightest possible intervals in cases 3.a and 3.b. However, since in these cases the function is very flat in that neighborhood, one does not really need a very tight bound on the zeros of the derivative. In [13] Dussel and Schmitt describe how one can use an interval version of the bisection method to get tight bounds on the zeros even if the function is not monotone. One can use that procedure if one wishes to .

1.4.4 The Dissection Algorithm.

The algorithm we use is a Modified Regula Falsi algorithm (see Conte and de Boor [9]). We use two versions: one for functions that take interval values, and the other for functions that take real values. Both algorithms differ from each other by minor details.

We assume that we are given an interval $[a,b]$ such that $f(a) \cdot f(b) < 0$. At each step we have FA and FB such that $FA \cdot FB < 0$. We compute λ by: $\lambda = 1/(1-FA/FB)$. Clearly $0 \leq \lambda \leq 1$.

In order to avoid problems of convergence and overflow-underflow, if $\lambda < \delta$ we set $\lambda = \delta$ or if $1-\lambda < \delta$ we set $\lambda = 1-\delta$. δ is a given small number, $\delta > 0$. After we determine $0 < \lambda < 1$, we compute a point x in the interval (a,b) by $x = \lambda \cdot a + (1-\lambda) \cdot b$. If $0 \in f(x)$, then we have found a zero, otherwise depending on the sign of $f(x)$, we replace a or b by x and FA or FB by $f(x)$.

If a (or b) had been replaced in the previous iteration and is being replaced again we set FB = FB/2 (or FA = FA/2). The above step prevents the possibility of approaching the zero only from one side. If $b-a \leq \varepsilon$ then we are done. Otherwise, we compute new λ and so on... . In the case where f is an interval valued function, If $0 \in f(x)$, then we check if $0 \notin f(x - \frac{\varepsilon}{2})$ and $0 \notin f(x + \frac{\varepsilon}{2})$; if this is true we take $(x - \frac{\varepsilon}{2}, x + \frac{\varepsilon}{2})$ to be the desired interval.

If $0 \in f(x - \frac{\varepsilon}{2})$ or $0 \in f(x + \frac{\varepsilon}{2})$ then we use the same procedure for $\bar{f}$ and $\underline{f}$ in order to find the intervals containing zero of $\bar{f}$ and $\underline{f}$. Using these intervals and the sign of f' we determine an interval containing the zero of f .

In case $\bar{f}(x) = 0$, $[x^-, x^+]$ is an interval containing a zero of $\bar{f}$ and the same is true for $\underline{f}$. (x^- is the biggest machine number smaller than x and x^+ is the smallest machine number greater than x). The above algorithm is linearly convergent since the interval is reduced at least by a factor of $1 - \delta$ at each iteration and at most by a factor of δ. However, if δ is small (say $\delta = 10^{-2}$ or 10^{-3}) the algorithm behaves like a quadratically convergent algorithm. In some sense this algorithm is in between the Bisection method ($\delta = \frac{1}{2}$) and the modified Regula Falsi ($\delta = 0$).

CHAPTER 2

A POSTERIORI ERROR BOUNDS - THE IMPLEMENTATION

2.1 On the Numerical Procedures.

In this section we describe the numerical procedures we use in order to compute upper bounds on the constants of Kantorovich's theorem.

Let us assume that we are given a numerical solution to equation (1) computed at a sequence of points $a = x_1 < x_2 < \ldots < x_{n+1} = b$. As an initial guess we take the piecewise polynomial function defined as follows: On each subinterval $[x_i, x_{i+1}]$, $i = 1,\ldots,n-1$ $x_0(t)$ is the sixth order polynomial that interpolates the solution and its derivative at the points x_i, x_{i+1}, x_{i+2}. The values of the derivatives are computed via the differential equation. On the interval $[x_n, x_{n+1}]$, $x_0(t)$ is the sixth order polynomial interpolating the solution and it's derivative at x_{n-1}, x_n, x_{n+1}.

First the residual is computed, that is:

$$r(t) = x_0(t) - x_0(a) - \int_a^t f(s, x_0(s))ds.$$

On each subinterval $x_0(t)$ is a polynomial. Since $f(s,y)$ is a polynomial in s and y, on each of the subintervals $r(t)$ is also a polynomial. Using the polynomial coefficients of $r(t)$ and the algorithm described in section (1.4.2) a bound on $\|r\|$ is computed.

Second, the fundamental and inverse fundamental solutions are numerically computed. One has to solve $2n$ Initial Value Problems: n for $Y(t)$ and n for $(Y^{-1}(t))^T$. The values of $Y(t)$ and $(Y^{-1}(t))^T$ are computed at the same sequence of points that $x_0(t)$

was computed.

We take $V(t)$ to be the piecewise polynomial function approximating $Y(t)$, and $W(t)$ to be the piecewise polynomial function approximating $Y^{-1}(t)$. $V(t)$ and $W(t)$ are computed with the same interpolation scheme used to construct $x_0(t)$.

The algorithm described in section (1.3.2) is used to compute bounds on $\|E_1\|$, $\|E_2\|$, $\|Y\|$, $\int_a^b |Y^{-1}(s)'|ds$ and $|E_1(1)|$.

The bounds on $\|x_1\|$, $\|x_2\|$ (of equation 13a, 13b) as well as the bounds on $\|V\|$, $\|W\|$, $\|W'\|$ and $\|A\|$ are computed using the rough bound described in section (1.3.2).

Third, $\tilde{R} = B_1 + B_2 V(b)$ and $C \approx R^{-1}$ are computed. Using the bounds on $|E(1)|$ and $|C|$ bounds on $|\tilde{R}^{-1}|$, $|\tilde{R}^{-1}-C|$ $\max(|R^{-1}B_1|, |R^{-1}B_1-I|)$ and $|R^{-1}B_2|$ are computed (see section (1.3.3)).

The facts that the problem has separated boundary conditions and that

$$B_1 = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & 0 \\ I & 0 \end{pmatrix}$$

implies that:

a) $|B_1| = |B_2| = 1$

b) if $Y(b) = \begin{pmatrix} \alpha & \beta \\ r & \delta \end{pmatrix}$,

where $\alpha, \beta, \gamma, \delta$ are $n/2 \times n/2$ matrices, then

$$R = \begin{pmatrix} I & 0 \\ \alpha & \beta \end{pmatrix} \text{ and } R^{-1} = \begin{pmatrix} I & 0 \\ -\beta^{-1}\alpha & \beta^{-1} \end{pmatrix},$$

$$R^{-1}B_1 = \begin{pmatrix} I & 0 \\ -\beta^{-1}\alpha & 0 \end{pmatrix}, \quad R^{-1}B_1 - I = \begin{pmatrix} 0 & 0 \\ -\beta^{-1}\alpha & 0 \end{pmatrix}$$

and $R^{-1}B_2 = \begin{pmatrix} 0 & 0 \\ -\beta^{-1} & 0 \end{pmatrix}$.

Therefore, $\max(|R^{-1}B_1|, |R^{-1}B_1-I|) = |\beta^{-1}\alpha| + 1$ and $|R^{-1}B_2| = |\beta^{-1}|$. Using the above bounds the bound on $\| F'(x_0)^{-1}\|$ is computed by formula (12).

The bound on $\| x_0-x_1\|$, η , is given by

$$\eta = \| Y\| \left(\int_a^b |Y^{-1}(s)|ds - (|\beta^{-1}\alpha| + 1) + |\beta^{-1}|\right)\| r\| + \| r\|.$$

Since the problem has simple boundary conditions one makes sure the initial guess $x_0(t)$ satisfies the boundary conditions. Therefore, one can assume $\rho = 0$. The above bound for η is smaller than the bound $\| F'(x_0)^{-1}\| \cdot \| r\|$.

2.2 Function Representations.

The numerical solution to equation (1), the fundamental solution and the inverse fundamental solution are continuous functions. However, these functions have a discrete representation inside the computer. We are using two types of representations:

a) Point-value representation: In this representation the function is represented by a sequence of points (the knot sequence) $a = x_1 < x_2 < \ldots < x_n = b$ and a sequence of values $y_1,\ldots,y_n$ which are the double precision values of the function at $x_1,\ldots,x_n$. We use the same knot sequence to represent $x_0(t)$, the approximate solution of equation (1), $V(t)$, the approximate fundamental solution and $W(t)$, the approximate inverse fundamental solution.

b) Piecewise polynomial representation: In this representation the function is represented by the knot sequence and a $6\times n$ array of numbers. On each subinterval $[x_i,x_{i+1}]$, the function is a 6th order polynomial $p(t)=\sum_{j=0}^{5} a_{j,i}(t-x_i)^j$. This polynomial interpolates the values of the function and its derivative at the points

x_i, x_{i+1}, x_{i+2}.

Since computers compute with a finite, discrete set of numbers, it is impossible in general, to compute the exact coefficients of the polynomials. Moreover, the coefficients, usually, are not machine representable numbers. On the other hand the initial guess $x_0(t)$, $V(t)$ and $W(t)$ are to be continuous functions. However, one cannot guarantee that the computed functions are continuous. To overcome this diffuculty we are using interval arithmetic (see [30]). Instead of computing with one floating point number the computation is done with two. One of the numbers is guaranteed to be smaller or equal to the number we need and the other, greater or equal to it. Therefore, one actually has two sets of polynomial coefficients. The first is a machine representable set of numbers. Each of them is guaranteed to be greater or equal to the corresponding exact coefficient. Similarly each floating point number in the other set is guaranteed to be smaller or equal to the corresponding exact coefficient. Therefore, one has two polynomials; one which at any point is greater or equal to the exact polynomial, and the other is smaller or equal to it. Although it is impossible to compute the exact polynomial itself, by bounding the range of these two polynomials one gets a bound on the range of the exact polynomial. (See also section 1.4.)

2.3 The Interval Arithmetic package.

As was said before we use interval arithmetic in order to take the effect of round-off error into account.

The interval arithmetic package we have constructed is based on the Multiple Precision Package (See Crary [10]). The interval precision package was designed to be used with the AUGMENT precompiler. All subroutines that use Interval or Multiple arithmetic are written in "extended" FORTRAN (See [11]), and are translated by AUGMENT to standard FORTRAN. For the use of AUGMENT see Crary [11]. The package is similar to the one written by Yohe [44]. However, the package is a "scaled down" version; only routines that are needed were constructed. Yohe's package was not used because we needed more precision than that package provides. Yohe's package is written in single precision arithmetic (approximately 8 decimal digits). Our package uses 4 words per floating point number. (About 32 decimal digits.) Therefore, each interval number takes 8 words.

The subroutines and functions available are:

1) The arithmetic operations: +, -, *, / .

2) Comparison operators: .LT. , .LE., .GT., .GE., .EQ., .NE..

3) Functions: INF, SUP, ABS, TSIGN, COMPOS, MAX. INF, SUP, ABS and COMPOS are the usual ones TSIGN is defined by:

$$\mathrm{TSIGN}([a,b]) = \begin{cases} 1 & \text{if} \quad 0 < a \\ 0 & \text{if} \quad a \leq 0 \leq b \\ -1 & \text{if} \quad b < 0 \end{cases}$$

and MAX is defined by:

$$MAX([a,b],[c,d]) = [f,f] \quad \text{where} \quad f = \max(b,d).$$

4) Conversion functions:

There are conversion functions from Integer, Double Precision, and Multiple to Interval and from Interval to Double percision and Integer. Type conversion has to be explicitly invoked and is not done automatically. That is, the statement AI = D where AI is of type Interval and D is Double precision, is not legal. The function CTX(·) converts its argument to Interval, CXTD converts Interval to double precision and CXTI converts Interval to Integer. Since the Multiple precision package is used to construct the interval package the description deck of the multiple package has to be submitted to AUGMENT in front of the Interval description deck. The description decks of Multiple and Interval are given in the Appendix.

2.4 The Subroutine Package.

2.4.4 General Information.

This package of subroutines enables one to bound $\| x_1 - x_0 \|$ and $\| F'(x_0)^{-1} \|$. Most of the computation is done in multiple and interval arithmetic. However, the user does not have to be very knowledgeable about the inner working of the multiple and interval packages. The main subroutine APOSTR does not have multiple or interval arguments in its calling sequence.

There are five, user supplied, problem dependent subroutines. Two subroutines out of the five use Interval arithmetic. In order to be able to write these routines the user has to know how to use AUGMENT (see [11]) and how to use a few polynomial manipulation subroutines. The polynomial subroutines, the interval package and the description decks are provided with the package. The underlying arithmetic is provided by the multiple precision package and is available on MRC*LIB., that is Mathematics Research Center's relocatable library. Since the multiple package is written in UNIVAC 1100 series assembler, the package is not portable and no attempt was made to make it portable. We tried to make the package applicable to a large set of problems. However, the package was designed as an experimental tool to test some of our ideas. It is not and was not meant to be a general purpose production code.

We now give a short description of the subroutines in the package. A complete listing of the subroutines, the extended FORTRAN source code as well as the Standard FORTRAN code produced by

AUGMENT is given as an appendix on a microfiche card.[†]

2.4.2 Problem Independent Subroutines.

APOSTR: This is the main subroutine. It sets up the work space for each of the subroutines, calls the subroutines that perform the different parts of the algorithm and computes the final bounds. See calling sequence description and explanations in the next section.

INLIZ: Initialization routine sets up some needed constants.

SETKNT: This subroutine converts the double precision knot sequence to interval valued knot sequence.

GREENF: This subroutine puts the values of the solution into common block /SPLCFF/ and also computes the values of $Y(t)$ and $(Y^{-1}(t))^T$ at the knot sequence. It uses subroutine DDESUB in order to solve the initial value problems.

CMPORS: This subroutine computes a bound on the norm of the residual, that is: A bound on $\| x_0(t) - \int_a^t f(s,x_0(s)ds - x_0(a) \|$.

BDGRFC: This subroutine implements the algorithm described in section (1.3.2). It computes bounds on $\|Y(t)\|$, $\int_a^b |Y^{-1}(s)'|ds$, $\|A\|$, $\|E_1\|$, $\|E_2\|$ and $|E_1(b)|$.

CMRINV: This subroutine computes an approximation to R^{-1} where $R = B_1 + B_2Y(b)$. The approximation is first computed by Gauss elimination and then improved by Newton's method. The subroutine computes bounds on $\|R^{-1}\|$, $\|B_1R^{-1}\|$ and $\|B_2R^{-1}\|$ using the method described in section (1.3.3)(see also section (21)).

[†]Only the MRC report has the microfiche card for obvious reasons. A copy of the microfiche card can be obtained from the author.

LEQRMR: This subroutine computes the residual of V, that is: $V(t) - \int_a^t A(s)V(s)ds - V(a)$. (Recall that $V(t)$ is the piecewise polynomial approximation to $Y(t)$.) LEQRMR is used by subroutine BDGRFC.

LQERMR: This subroutine computes the residual of W, that is $W(t) - W(a) + \int_a^t W(s)A(s)ds$. (Recall that $W(t)$ is the piecewise polynomial approximation to $Y^{-1}(t)$). LQERMR is used by BDGRFC.

CBSTPP: This subroutine converts point value representation of a function to its piecewise polynomial representation. It computes the coefficients of the polynomial interpolating the function and it's derivative at 3 consecutive points.

PPNRM: This subroutine computes a tight bound on the norm of a polynomial in a given interval. It uses subroutine PNORM which is the implementation of the algorithm described in section (1.4.2).

PNORM: This subroutine implements the algorithm described in section (1.4.2). It isolates the zeros of the polynomial's derivative inside very small intervals and then bounds the norm of the polynomial. It uses subroutine LOCATE in order to bracket the zeros of the polynomial's derivatives and subroutine NORM1 in order to compute bounds on the norm of the polynomial.

NROM1: This subroutine is given a sequence of intervals containing the zeros of the polynomial's derivative using this information it bounds the polynomial's norm.

LOCATE: This subroutine is part of the algorithm for getting a tight bound on the norm of a polynomial. This subroutine gets a sequence of points where the kth derivative of the polynomial is either positive or negative or may have a zero. Employing the method described in section (1.4.3) subroutine LOCATE computes a sequence of points where the (K-1)th derivative is positive or negative or may have a zero. The intervals that could possibly contain a zero are made smaller than a given tolerance.

BISECT: This is the dissection algorithm described in section (1.4.4). This subroutine brackets the zeros of sup or inf of an interval valued polynomial. The polynomial is assumed to have a zero in the given interval. It is also assumed to be monotone.

BSCINT: The same as BISECT only it is the interval version of the dissection algorithm.

PMXNRM: This subroutine computes a bound on the uniform norm of a polynomial matrix. The bound is given by: $\max\limits_{1 \le i \le n} \sum\limits_{j=1} \| a_{ij} \|$. The bounds on the terms $\| a_{ij} \|$ are computed by the "rough bound" method described in section (1.4.1).

UBEXP: This subroutine computes a bound on e^a where a is a positive interval number.

VCTINT: This subroutine integrates piecewise polynomial functions. That is: it is given $\int_a^{x_i} u(s)ds$ and the polynomial coefficients of $u(t)$, $x_i \le t \le x_{i+1}$, and it computes the polynomial coefficients of $\int_a^t u(s)ds$, $x_i \le t \le x_{i+1}$. u is a vector valued function.

PPVLU1: This subroutine computes the double precision value of the solution at any given point.

DIVDIF: Divided differences subroutine. It is used by PPVLU1 to compute the interpolating polynomials.

IDVDIF: The same as DIVDIF only in interval arithmetic. It is used by CBSTPP in order to compute the polynomial's coefficients.

DDESUB: This subroutine solves initial value problems in double precision. This subroutine uses Gragg's modified midpoint rule with Bailirsch-Store's rational extrapolation method. It is essentially the same subroutine as the one provided by Madison Academic Computing Center. A modification was made to enable the routine to store the solution's values at any given sequence of points not necessarily equally spaced. For details about the use of the subroutine see [49].

2.4.3 Polynomial Manipulation Subroutines.

In order to facilitate the manipulation of polynomials several subroutines were constructed. All polynomials have interval coefficients. The following subroutines are provided:

SUBROUTINE PLYNEG(K,A,B)

This subroutine computes B = -A, A and B are Kth order polynomials.

SUBROUTINE PLYADD(K,A,B,C)

This subroutine computes C = A+B, A,B and C are Kth order polynomials.

SUBROUTINE PLYSUB(K,A,B,C)

This subroutine computes C = A-B, A, B, and C are Kth order polynomials.

SUBROUTINE IPYMUL(K1,K2,A,B,C)

This subroutine computes $C = A \times B$ A is K1th order polynomial, B is K2th order polynomial and C is (K1+K2-1)th order polynomial.

SUBROUTINE PLYDRV(K,A,B)

This subroutine computes $B = A'$, A is Kth order polynomial.

2.4.4 Problem Dependent Subroutines.

There are five problem dependent, user supplied subroutines. Two of the subroutines compute with interval arithmetic and three with double precision.

SUBROUTINE FUNC(XI,SOL,FSOL,L,K,KF)

This subroutine H is given the polynomial representation of $x_0(t)$ on the interval $[xI(1),xI(2)]$ and it computes the polynomial representation of $f(t,x_0(t))$. The solution's polynomials are Kth order and are interval valued.

The parameters:

xI:- Interval valued vector of dimension 2. It gives the end points of the interval on which the polynomial representation is valid.

SOL:- $K \times L$ interval array with the coefficients of the polynomials.

FSOL:- $KF \times L$ array with the coefficients of $f(t,x_0(t))$.

L:- The size of the system.

KF:- The first dimension of FSOL.

SUBROUTINE SETA(K,L,XI,SOL,A,KA)

This subroutine is given the polynomial representation of the solution and it computes the polynomial representation of $A(t) = f_x(t,x_0(t))$.

The parameters:

K,L,XI,SOL:- The same as in FUNC

A:- A $KA \times L \times L$ interval array with the coefficients of A(t).

Remarks.

1) The above subroutines should be translated with AUGMENT.

2) The user could use the polynomial manipulation subroutines to compute $f(t,x_0(t))$ and $A(t)$.

3) Note that the polynomials are of the form

$$P(T) = \sum_{J=1}^{K} D(J)*(T-XI(1))**(J-1)$$

Therefore, the polynomial representation of the independent variable t on the interval [XI(1), XI(2)] is (XI(1),1,0,...,0).

SUBROUTINE DEFUNC(T,Y,DY,STOR,IFLAG)

This subroutine is given the value of the solution at T in Y and it computes f(T,Y) and stores it in DY. This subroutine is used in computing the piecewise polynomial representation of the solution.

The parameters:

T:- Point of evaluation.

Y:- L dimensional double precision vector.
The value of the solution at T.

DY:- L dimensional double precision vector contains the values of f(T,Y).

STOR:- Not used.

IFLAG:- Not used.

SUBROUTINE DERIVS(T,Y,DY,STOR,IFLAG)

This subroutine is used in order to compute the fundamental solution. It is given in Y the values of one column of $Y(t)$ and in DY the values of $A(t)Y$.

<u>The parameters</u>:

T,Y,DY:- The values of Y(I), I = 1,...,L have to be put in STOR (See [49] for saving options).

IFLAG:- not used.

SUBROUTINE TRDERV(T,Y,DY,STOR,IFLAG)

This subroutine is used in order to compute $(Y^{-1}(t))^T$. It is given in Y the values of one column of $(Y^{-1}(t))^T$ and stores in DY the vector $-A^T(t)Y$.

The parameters are the same as in DERIVS.

The differential equations that $Y(t)$ and $(Y^{-1}(t))^T$ satisfy, depend on the solution $x_0(t)$. Therefore, the values of the solution at the knot sequence are put into an array in the common block /SPLCFF/LXI,C(·). LXI is the number of points the solution is computed at and C is LXI×L double precision array with the solution values at the knot sequence.

In order to facilitate the computation of $x_0(t)$ at any point a subroutine is provided:

SUBROUTINE PPVLU1(C,LXI,YHT,X)

<u>The parameters</u>:

LXI:- Number of points in the knot sequence.

C:- LXI×L array with the values of the solution at the knot sequence.

L:- The size of the system.

YHT:- L dimensional array with the values of the solution at x.

X:- Point of evaluation.

2.4.5 The Subroutine APOSTR.

The main subroutine is APOSTR. It's calling sequence is:

SUBROUTINE APOSTR(KA,KF,L,NPT,SOLTN,FUNC,SETA,B1,B2,NRMINV,ETANRM, WRKSPC).

The parameters:

KA:- The order of the polynomials in the matrix $A(t) = f_x(t,x_0(t))$.

KF:- The order of the polynomials in $f(t,x_0(t))$ (Remember that $x_0(t)$ is 6th order piecewise polynomial function).

L:- The size of the system of differential equations.

NPT:- The number of points in the knot sequence.

SOLTN:- NPT×L double precision array with the values of the solution $x_0(t)$ at the knot sequence.

FUNC:- Problem dependent subroutine.

This subroutine is given the polynomial representation of $x_0(t)$ and it computes the polynomials $f(t,x_0(t))$ for calling sequence. See section (2.4.4).

The subroutine FUNC should be declared external in the calling program.

SETA:- Problem dependent subroutine.

This subroutine is given the polynomial representation of $x_0(t)$ and it computes the matrix polynomial $A(t) = f_x(t,x_0(t))$. For calling sequence see (2.4.4). This subroutine should be declared EXTERNAL in th calling program.

B1,B2:- L×L double precision arrays with the coefficients B_1 and B_2.

NRMINV:- Double precision value of the bound on $\| F'(x_0)^{-1} \|$

ETANRM:- Double precision value of the bound on $\| x_1 - x_0 \|$.

WRKSPC:- Double precision work space array.

The dimension of WRKSPC should be at least 2*NPT*(L*L+2)+4*L*L*(38+3*KA).

Please note

The knot sequence is not part of the calling sequence. It must be put into the common block /DPSTEP/ STEPS(·) prior to the call to APOSTR.

2.5 Examples.

In this section we give two examples of computational existence proofs. Analytical existence proofs for these problems are not known.

2.5.1 Example 1.

Consider the following two-point boundary value problem:

$$\varepsilon y'' = (y^2 - (t-1)^2)y' \qquad 0 \le t \le 1 \qquad \text{(E1)}$$

$$y(0) = A\ , \quad y(1) = B\ .$$

The above problem was suggested and analyzed by Howes and Parter as a model problem for nonlinear problems having a continuous locus of singular points (see [17]). They studied the asymptotic behaviour of solutions to this problem as $\varepsilon \to 0^+$. It is not hard to show that the above problem has at least one solution for any value of $\varepsilon > 0$. However the existence of multiple solutions was not ruled out. Moreover, Howes and Parter showed that if $0 < B \le \frac{1}{\sqrt{3}}$, $\frac{1}{\sqrt{3}} \le A \le 1$ there are at most three limit solutions as $\varepsilon \to 0^+$; $y \equiv A$, $y \equiv B$ and $y \equiv \frac{1}{\sqrt{3}}$. Indeed Francis Sutton [36] computed, using finite differences, three solutions for $\varepsilon > 0$ small, thus implying that all three possible limit solutions are in fact obtained.

We also have found three distinct numerical solutions in that range. Our numerical solutions were obtained by a collocation method. The subroutine LOBATO by deBoor and Weiss [4] was used to obtain the numerical solutions. Taking $A = .96$, $B = .001$ and $\varepsilon = 1/15$ the above aposteriori error analysis was used to establish the existence of (at least) three distinct solutions.

Moreover, guaranteed error bounds were obtained.

Of course, the theory does not rule out the existence of more solutions for $\varepsilon > 0$. However, the theoretical results of Howes and Parter combined with Sutton's computational results and our existence results for $\varepsilon = 1/15$ seem to support Sutton's conjecture that Equation (El) has exactly 3 solutions for all $0 < \varepsilon < \varepsilon_0$, for some $\varepsilon_0 > 0$.

Although $\varepsilon = 1/15$ is not very small Equation El is already "stiff". As a matter of fact solution 3 (see figure 2) was "stiffer" than solutions 1 and 2. In order to get reasonable error bounds we had to modify the original programs so that it would be possible to subdivide the interval into unequal subintervals. The bounds on the residual were reduced by 4-5 orders of magnitude by putting more points in the interval where the function and it's derivative change very fast. Solutions 1 and 2 were computed using 201 points and solution 3 using 301 points. The bounds we obtained are given in Table 1. We now turn to our computations:

Let us rewrite equation (El) as a first order system with $R = 1/\varepsilon$.

$$\frac{d}{dt}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ R(y_1^2 - (t-1)^2)y_2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} y_1(1) \\ y_2(1) \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix}$$

the matrix of partial derivatives is given by:

$$A(t) = \begin{pmatrix} 0 & 1 \\ 2Ry_1y_2 & R(y_1^2 - (t-1)^2) \end{pmatrix}$$

The second partial derivatives are:

$$\frac{\partial^2 f_1}{\partial y_1 \partial y_2} = \frac{\partial^2 f_1}{\partial^2 y_1} = \frac{\partial^2 f_2}{\partial^2 y_2} = 0$$

and

$$\frac{\partial^2 f_2}{\partial^2 y_1} = 2Ry_2 , \quad \frac{\partial^2 f_2}{\partial^2 y_2} = 0, \quad \frac{\partial^2 f_2}{\partial y_1 \partial y_2} = 2ry_1$$

therefore

$$\sup_{\|x-x_0\| \le r} \| F''(x) \| \le 2R(2\int_0^1 |y_1(s)|ds + \int_0^1 |y_2(s)|ds) + 6\,R\,r\,.$$

Therefore, the bound on the second Frechet derivative can be computed by the above formula. Since y_1 and y_2 do not change sign on $[0,1]$ the above integrals can be computed exactly (Remember that y_1 and y_2 are piecewise polynomial functions.).

In figure 2 we have plotted the three different solutions.

The error bounds we obtained are:

	K	η	B	h	r_0
1	77.5179	2.40683×10^{-9}	4.4769×10^{2}	8.3526×10^{-5}	2.4069×10^{-9}
2	60.9082	1.24323×10^{-9}	5.0692×10^{2}	3.8386×10^{-5}	1.24326×10^{-9}
3	37.98871	5.39276×10^{-9}	1.7473×10^{3}	3.5795×10^{-4}	5.3937×10^{-9}

Table 1

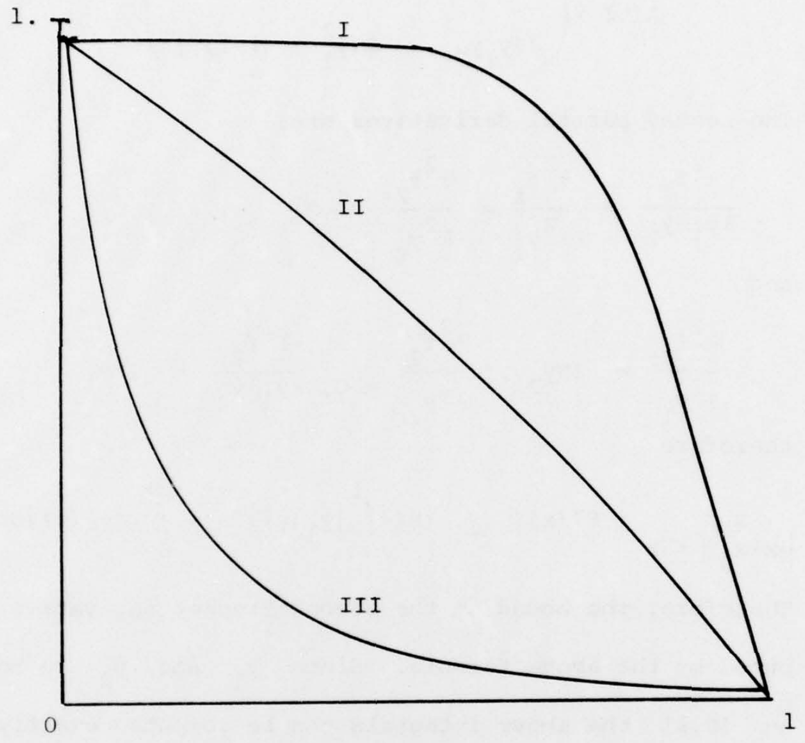


Figure 2

2.5.2 Example 2.

This example was mentioned in the introduction. These differential equations describe the flow between two parallel infinite disks rotating about a common axis. The equations are:

$$h^{iv} + h\,h''' + g\,g' = 0 \tag{E2}$$
$$g'' + h\,g' - h'g = 0$$
$$h(0) = h'(0) = h(1) = h'(1) = 0$$
$$g(0) = \Omega_0 \ , \quad g(1) = \Omega_1 \ .$$

Although the above problem has attracted considerable attention (See McLeod [22] and the references there), existence proofs for solutions outside a small set of values of (Ω_0, Ω_1) are not known (see Elcrat [14] and McLeod [22]).

Hastings proved existence provided Ω_0 and Ω_1 are sufficiently small.

Elcrat, using a fixed point argument proved that: If

a) $\Omega_1 \in [-\Omega_0, 0]$ and $0 \leq \Omega_0 \leq C$

or

b) $\Omega_1 \in (0, \Omega_0]$ and $\Omega_0^2 - \Omega_1^2 < C^2$,

where $C = \frac{4}{3}(2)^{.5}(\exp(.25) + \exp(\frac{4}{9}) - 1)^{-1/2} \approx 1.5$ then equation (E2) has a solution.

McLeod and Parter [28] proved existence of solutions for the counter rotating case $(\Omega_0 = -\Omega_1)$. They proved existence of solutions for all values of $\Omega_0 > 0$. Proof of existence in all other cases is not known.

Many people have computed solutions outside the range where the existence of solutions is known. Moreover, multiple numerical solutions were obtained. However, none of the computational papers give any error analysis. Although many of these computations probably give reasonable approximation to solutions of equations (E2), there is at least one case where a computed solution was proven to be incorrect. (See McLeod and Parter [28].)

By using an a posteriori error analysis, the existence of a solution, outside the range where the previously known results apply, has been guaranteed. Although we have not proven the existence of multiple solutions (when Ω_0 and Ω_1 are large enough), we believe that with enough time and computing power (money) one could use our method to establish the existence of multiple solutions. As a by-product of Elcrat's proof the solutions he obtains are "monotone" in the sense that g' is of one sign. On the other hand, McLeod and Parter showed that if there is a solution in the case where Ω_0 and Ω_1 are: positive, large enough and far apart, g' of that solution must change sign. Moreover, numerical solutions where g' does change sign were obtained (for example see Cerutti [6]). It will be interesting to prove the existence of such solutions.

Note that the above problem is "stiff" and it gets "stiffer" as $|\Omega_0|$ and $|\Omega_1|$ grow.

This numerical solution was computed with subroutine LOBATO at 101 equally spaced points. We computed with $\Omega_0 = 7$ and $\Omega_1 = 1$. Note that $\Omega_0^2 - \Omega_1^2 = 48$.

The bounds we obtained are:

K	η	B	h	r_0
4	2.0154×10^{-6}	1.60437×10^{4}	.1293406	2.16597×10^{-6}

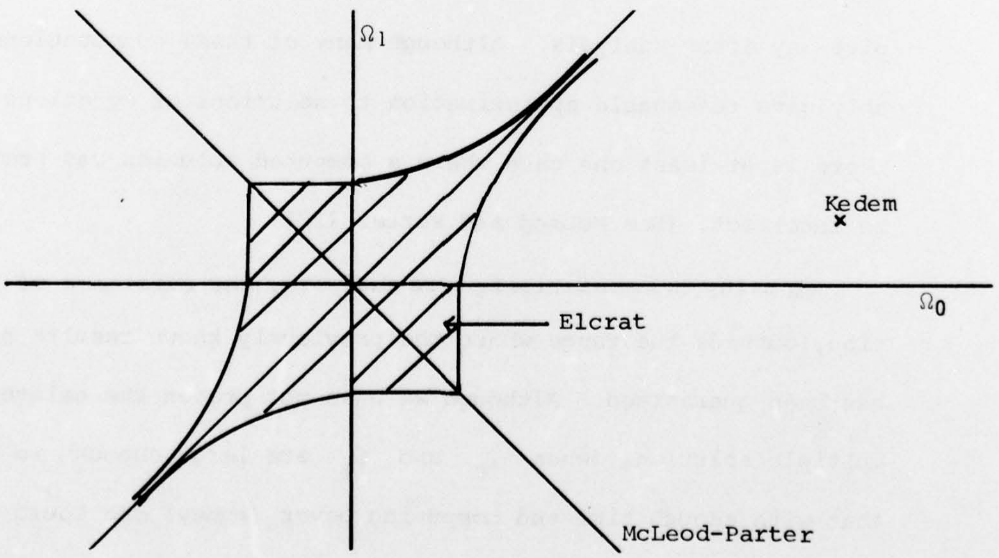


Figure 3

In Figure 3 we have plotted the values of Ω_0 and Ω_1 for which existence proofs are known.

2.6 Conclusions, Remarks, etc.

In this work we have demonstrated a way to compute aposteriori error bounds for polynomials two-point boundary value problems. Some of the difficulties of other methods were overcome. However, the problem is by no means completely solved. We list below problems and questions left to be answered:

a) The bound on $\eta = \| x_1 - x_0 \|$ we use, is not the best possible one. We suggested two other methods for computing η. These methods have to be further investigated. Both methods suffer from some difficulties. The first method uses the explicit form of the Green's function and requires the manipulation of high degree polynomials. The second method looks more promising, however, a large amount of work may be necessary in order to make the residual, that is $x_1(t) - \int_a^t [A(s)x_1(s) - r(s)]ds$ very small. In any case more experiments are needed before one could obtain better methods for aposteriori error bounds.

b) We have chosen to interpolate the solution $x_0(t)$, $Y(t)$ and $Y^{-1}(t)$ by 6th order polynomials. However, there are no compelling reasons for doing so. More experiments are needed to obtain some ideas and to gain some insight into good strategies for choosing interpolation schemes.

c) The implementation of interval arithmetic, we use, is very slow. A necessary condition for making interval arithmetic practical for routine use is having the arithemtic done by hardware and not by software. The prospects, in the near future, of having interval arithmetic as a standard hardware option are not good, However, a new generation of microprogramable computers may make the use of interval arithmetic practical.

d) We use interval arithmetic in order to take round-off error into account. If one wishes to prove the existence of solutions one is forced to take round-off error into account. However, it is our experience that computations with double precision arithmetic give the same results and round-off errors have negligible effects. Thus, our method could be used to compute reliable error bounds although the results would not be completely rigorous existence proof.

e) The size of the residual or the size of $\|x_1-x_0\|$ on each subinterval could be used to decide how to redistribute or refine the subdivision points. More experiments are needed in order to find strategies based on the above information. Also one should compare these methods with other methods suggested in the literature (see [2], [7], [45], [48]).

f) As was said in the introduction our method could be extended to problems that are not polynomials. If the function $f(t,y)$ of equation (1) is a factorable function of t and y (see Part A of our thesis), then, since $x_0(t)$ is a piecewise polynomial function, $f(t,x_0(t))$ and $A(t) = f_x(t,x_0(t))$ are piecewise factorable functions. Since one knows the Taylor series expansion of $x_0(t)$ on each subinterval, one can compute Taylor series approximations to $f(t,x_0(t))$ and $A(t)$.

If $\tilde{f}(t)$ is the approximation to $f(t,x_0(t))$ and $\tilde{A}(t)$ is the approximation to $f_x(t,x_0(t))$, then:

i) Using interval techniques (see Moore [30] ch. 10) one can bound $\|\tilde{f}(t)-f(t,x_0(t))\|$ and $\|A(t) - \tilde{A}(t)\|$. Also if one takes enough terms in the series expansion and the subintervals are small enough, these bounds will be small.

ii) Since $\tilde{f}(t)$ is a piecewise polynomial function we can use it in order to compute an approximation to $r(t)$. Recall that $r(t) = x_0(t) - x_0(a) - \int_a^t f(s,x_0(s))ds$. If $\tilde{r}(t) = x_0(t) - x_0(a) - \int_a^t \tilde{f}(s)ds$ then $r(t) - \tilde{r}(t) = -\int_a^t (f(s,x_0(s)) - \tilde{f}(s))ds$ therefore

$$\|r\| \le \|\tilde{r}\| + (b-a) \cdot \|f(t,x_0(t))-\tilde{f}(t)\| .$$

iii) Similarly, $\tilde{A}(t)$ can be used to compute approximation to $X_1(t)$ and $X_2(t)$.

Recall that

$$X_1(t) = V(t) - V(a) - \int_a^t A(s)V(s)ds .$$

If

$$\tilde{X}_1(t) = V(t) - V(a) - \int_a^t \tilde{A}(s)V(s)ds$$

then

$$\|X_1-\tilde{X}_1\| \le \|A-\tilde{A}\| \cdot (\int_a^b |V(s)|ds) .$$

The same way, if $\tilde{X}_2(t) = W(t) - W(a) + \int_a^t W(s)\tilde{A}(s)ds$ then

$$\|X_2-\tilde{X}_2\| \le \|A-\tilde{A}\| \cdot (\int_a^b |W(s)|ds) .$$

In turn the bounds on $\|X_1\|$ and $\|X_2\|$ can be used to compute bounds on $\|E_1\|$ and $\|E_2\|$. (See section (1.3.2)).

Therefore, an algorithm, similar to the one used for polynomial equations, can be used for aposteriori error analysis of factorable two-point boundary value problems.

g) It is not hard to see how one can extend our method for problems *with nonlinear boundary conditions*.

REFERENCES

1. I. Babuška and W. C. Rheinboldt. A Posteriori Error Estimates for the Finite Element Method. Comp. Sci. Tech. Report TR-581. Univ. of Maryland, September 1977.

2. C. deBoor and B. Swartz. Collocation at Gaussian Points. SIAM J. Numer. Anal. Vol. 10, No. 4, September 1973, 586-606.

3. C. deBoor. Good approximation by splines with variable knots, II. Conference on the Numerical Solution of Differential Equations, Lecture Notes in Mathematics, Springer, Vol. 363, 1973.

4. C. deBoor and R. Weiss. SOLVEBLOK: A Package for Solving Almost Block Diagonal Linear Systems, With Applications to Spline Approximation and the Numerical Solution of Ordinary Differential Equations. Math. Res. Center TSR #1625. Univ. of Wisconsin-Madison. May 1976.

5. J. H. Cerutti. Collocation for Systems of Ordinary Differential Equations. Comp. Sci. Dept. University of Wisconsin-Madison Tech. Report #230, December 1974.

6. J. H. Cerutti; High Reynolds Number Flow Between Rotating Co-axial Disks; A Numerical Experiment. Comp. Sci. Dept. University of Wisconsin-Madison Tech Report #249, April 1975.

7. J. Christiansen and R. D. Russell. Error Analysis for Spline Collocation Methods with Application to Knot Selection, Math. of Comp., to appear.

8. E. A. Coddington and N. Levinson. Theory of Ordinary Differential Equations, McGraw-Hill, 1955.

9. S. D. Conte and C. deBoor. Elementary Numerical Analysis. An Algorithmic Approach. McGraw Hill 1972.

10. F. D. Crary. Multiple Precision Arithmetic Design With an Implementation on the UNIVAC 1108. Math. Res. Center TSR #1123. University of Wisconsin-Madison, May 1971.

11. F. D. Crary. The AUGMENT Precompiler I, User Information. Math. Res. Center TSR #1469. Univ. of Wisconsin-Madison, December 1976.

12. D. M. Cruickshank and K. Wright. Computable Error Bounds for Polynomial Collocation Methods, to appear in SIAM J. of Num. Anal.

13. R. Dussel and B. Schmitt. Die Berechnung von Schranken für den Werteberiech eines Polynoms in einem Intervall. Computing 6, (1970), 35-60.

14. A. R. Elcrat. On the Swirling Flow Between Rotating Coaxial Disks. Journal of Diff. Eq. 13, (1975), 423-430.

15. P. Hartman. Ordinary Differential Equations, Hartman 1973.

16. P. Henrici. Discrete Variable Methods in Ordinary Differential Equations.

17. F. A. Howes and S. V. Parter. A Model Nonlinear Problem Having a Continuous Locus of Singular Points. To appear in Studies in Appl. Math.

18. E. L. Ince. Ordinary Differential Equations. Dover, 1956.

19. E. Isaacson and H. B. Keller. Analysis of Numerical Methods. John Wiley and Sons, 1966.

20. L. V. Kantorovich and G. P. Akilov. Functional Analysis in Normed Spaces. Pergamon Press, 1964.

21. H. B. Keller. Numerical Methods for Two-Point Boundary-Value Problems. Blaisdell, 1968.

22. H. B. Keller. Approximation Methods for Nonlinear Problems with application to Two-Point Boundary Value Problems. Math. of Comp. Vol. 29, No. 130. April 1975, 464-474.

23. H. B. Keller. Numerical Solution of Two-Point Boundary Value Problems. SIAM Regional Conference Series in Applied Math. #24, 1976.

24. M. A. Krasnosel'skii, et. al. Approximate Solution of Operator Equations, Wolters-Noordhoff, Groningen, 1972.

25. J. N. Lyness and J. J. Kaganove. Comments on the Nature of Automatic Quadrature Routines, ACM-TOMS Vol. 2, No. 1, March 1976, 65-81.

26. M. A. McCarthy and R. A. Tapia. Computable A Posteriori L_∞ Error Bounds for the Approximate Solution of Two-Point Boundary Value Problems. SIAM J. Numer. Anal. Vol. 12, No. 6, December 1975, 919-937.

27. J. B. McLeod. Swirling Flow - A Survey. Math. Res. Center TSR #1473. Univ. of Wisconsin - Madison. January 1976.

28. J. B. McLeod and S. V. Parter. On the Flow Between Two Counter-Rotating Infinite Plane Disks. Arch. Rational Mech. Anal. 54, 4 (1974), 301-327.

29. R. E. Moore. On Computing the Range of a Rational Function of n Variables Over a Bounded Region. Computing 16, (1976) 1-15.

30. R. E. Moore. Interval Analysis, Prentice-Hall, 1966.

31. S. V. Parter. A Posteriori Error Estimates. Comp. Sci. Dept. University of Wisconsin Tech. Rept. #214. May 1974. In Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, A. K. Aziz, Ed. Academic Press, NY.

32. L. B. Rall. Computational Solution of Nonlinear Operator Equations. John Wiley and Sons, Inc., 1969.

33. L. B. Rall and R. A. Tapia. The Kantorovich Theorem and Error Estimates for Newton's Method. Math. Res. Center TSR #1043, Univ. of Wisconsin-Madison. February 1970.

34. S. M. Roberts and J. S. Shipman. The Kantorovich Theorem and Two-Point Boundary Value Problem. IBM J. Res. Develop. 10 (1966), 402-406.

35. R. D. Russell. Collocation for Systems of Boundary Value Problems. Numer. Math. 23 (1974), 119-133.

36. F. Sutton. Personal communication.

37. T. D. Talbot. Guaranteed Error Bounds for Computed Solutions of Nonlinear Two-Point Boundary Value Problems. Ph. D. Thesis, Computer Science Department, University of Wisconsin-Madison, 1968.

38. R. A. Tapia. The Weak Newton Method and Boundary Value Problems. SIAM J. Numer. Anal. Vol. 6, No. 4. December 1969.

39. R. A. Tapia. Newton's Method for Problems With Equality Constraints. SIAM J. Numer. Anal. Vol. 11, No. 1. March 1974.

40. G. M. Vainikko. Galerkin's Perturbation Method and the General Theory of Approximate Methods for Nonlinear Equations. USSR Comp. Math. Physics, 7 (1967), 1-41.

41. L. O. Wilson and N. L. Schryer. Flow Between a Stationary and a Rotating Disk With Suction. J. of Fluid Mech. to appear.

42. J. M. Yohe. Best Possible Floating Point. Math. Res. Center TSR #1054. Univ. of Wisconsin-Madison. March 1970.

43. J. M. Yohe. Software for Interval Arithmetic: A Reasonably Portable Package. Math. Res. Center TSR #1731. Univ. of Wisconsin-Madison. March 1977.

44. J. M. Yohe. The Interval Arithmetic Package. Math. Res. Center TSR #1755. Univ. of Wisconsin-Madison. June 1977.

45. A. B. White, Jr. On Selection of Equidistributing Meshes for Two-Point Boundary-Value Problems. To appear.

46. P. E. Zadunaisky. On the Estimation of Errors Propagated in the Numerical Integration of Ordinary Differential Equations. Numer. Math, 27, (1976), 21-39.

47. V. Pereyra. Variable Order Variable Step Finite Difference Methods for Nonlinear Boundary Value Problems. Conference on the Numerical Solution of Differential Equations, Dundee 1973. Lecture Notes in Math. 363. Springer 1974.

48. V. Pereyra and E. G. Sewell. Mesh Selection for Discrete Solution of Boundary-Value Problems in Ordinary Differential Equations, Numer. Math. 23 (1975), 261-268.

49. *Double Precision Differential Equation Package*, Reference Manual for the 1110 Academic Computing Center. Univ. of Wisconsin-Madison. August 1975.

50. J. B. McLeod and S. V. Parter. The Non-Monotonicity of Solutions in Swirling Flow. Math. Res. Center. TSR #1551. Univ. of Wisconsin-Madison. October 1975.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 1825 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A POSTERIORI ERROR BOUNDS FOR TWO-POINT BOUNDARY VALUE PROBLEMS		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Gershon Kedem		8. CONTRACT OR GRANT NUMBER(s) DAAG29-75-C-0024 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center/ University of Wisconsin, 610 Walnut Street, Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS #7 - Numerical Analysis
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709		12. REPORT DATE January 1978
		13. NUMBER OF PAGES 65
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Two-point boundary value problem, a posteriori, error bounds.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Consider a general Two-Point Boundary Value Problem (TPBVP):

$$y'(t) = f(t,y) \qquad a \le t \le b$$
$$B_1 y(a) + B_2 y(b) = w$$

where $f: R^{n+1} \to R^n$, $f \in C^2$, B_1 and B_2 are $n \times n$ matrices and $w \in R^n$.
It is shown how one can bound a posteriori the error made in the numerical solution of the (TPBVP). The error bounds obtained are rigorous and include the truncation and the roundoff error. In addition, the computations establish → next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

← Two-Point Boundary Value Problem

the existence of solutions to the TPBVP.

Numerical schemes are developed for the case where $f(t,y)$ is a polynomial in t and y. Examples are given of computational existence proofs for problems where analytical existence proofs are not known.